VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Využití prostředků SMath studio pro simulace zpracování signálů

SMath Studio in DSP

2017 Josef Nudera

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Josef Nudera**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2601R013 Telekomunikační technika

Téma: Využití prostředků SMath Studio pro simulace zpracování signálů
SMath Studio in DSP

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je popsat možnosti využití prostředí SMath Studio pro výuku zpracování signálů.

1. Popis instalace SMath Studio (Linux, Windows).
2. Popis práce s proměnnými typu "skalár", "vektor" a "matice (pole)" s ohledem na indexování jednotlivých prvků.
3. Popis možností grafického zobrazení výsledků výpočtů. 2D a 3D grafy.
4. Vytvoření vzorových výukových scriptů a funkcí z oblasti zpracování signálů (dle pokynů vedoucího práce).
5. Popis scriptů a vytvoření návod k jejich použití.
6. Subjektivní porovnání práce v prostředí SMath Studio s prostředím MATLAB.

Práce bude vytvořena v typografickém systému LaTeX.

Seznam doporučené odborné literatury:

[1] Bernard Liengme. *SMathPrimer*.[online] URL http://smath.info/?file=739837 [cit. 2013-02-11]

[2] ZAPLATÍLEK, K.; DOŇAR, B. *MATLAB pro začátečníky*. Vydání 2. Praha: BEN - technická literatura, 2005. 151 s. ISBN 80-7300-175-6.

[3] ZAPLATÍLEK, K.; DOŇAR, B. *MATLAB: začínáme se signály*. Vydání 1. Praha: BEN - technická literatura, 2006. 271 s. ISBN 80-7300-200-0.

[4] MACHÁČEK Z., NEVŘIVA P. *Modulované signály*. Vydání 1. VŠB - Technická univerzita Ostrava, 2012. ISBN 978-80-248-2600-4.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Skapa, Ph.D.**

Datum zadání: 01.09.2015

Datum odevzdání: 14.07.2017

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Souhlasím se zveřejněním této bakalářské práce dle požadavků čl. 26, odst. 9 *Studijního a zkušebního řádu pro studium v bakalářských programech VŠB-TU Ostrava.*

V Ostravě 14. července 2017 .....

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 14. července 2017 .....

Rád bych touto cestou poděkoval panu Ing. Janu Skapovi, Ph.D. za odborné vedení bakalářské práce a také za cenné rady, které mi pomohly tuto práci zkompletovat.

Abstrakt

Cílem bakalářské práce je seznámit uživatele s prostředím SMath Studia, freewarového programu. Tématem bakalářské práce je popis možností využití prostředí SMath Studia pro oblast zpracování signálů. SMath Studio umožňuje výpočty a jejich zobrazování do grafů. V rámci práce jsou také ukázány příklady práce v něm a použití tohoto programu v praktické úloze z oblasti zpracování signálů. Obsahuje vzorový výukový script a funkce z této oblasti s jeho popisem a návodem k jeho použití. Bakalářská práce by měla sloužit k výuce zpracování signálů. V závěru práce je subjektivní porovnání práce ve SMath Studiu oproti licencovanému prostředí Matlab.

Klíčová slova: SMath Studio; zpracování signálů; využití prostředí SMath Studio; výuka signálů; matice, 2D graf; 3D graf; propust; filtry; pásmová propust; pásmová zádrž; horní propust; dolní propust; aproximace; IIR

Abstract

Purpose bachelor's work is familiarise users with environment SMath studio's – freeware program. The theme of bachelor's work is description of options use of the environment SMath studio's for section signal processing. SMath studio allows for calculations and processing them into the graph. The work also banned examples of the use of the environment in practical problems from the area signal processing. It includes sample scripts and teaching functions of this area with their descriptions and instructions for their use. Thesis should be used to educate signal processing. The conclusion is subjective comparison work SMath studio trial against licensed Matlab.

Keywords: SMath studio; signal processing; use environment SMath studio; teaching signals; matrix; 2D graph; 3D graph; filters; bandpass filter; band stop filter; high pass filter; lowpass filter; approximation; IIR

Seznam použitých zkratek a symbolů

IIR	–	Infinite Impulse Response
ω_s, ω_p	–	transformace kmitočtů z digitální do analogové oblasti
$H_p(p)$	–	Frekvenční přenos $H_p(p)$
$H_p(\Omega)$	–	Zobrazení frekvenčního přenosu pro Ω
NDP	–	normovaná dolní prospust
$H_s(\omega)$	–	Zobrazení frekvenčního přenosu pro ω
$H_z(f)$	–	Frekvenční přenos
f_{vz}	–	Vzorkovací frekvence
f_N	–	Nyquistova frekvence
T_{vz}	–	Perioda vzorkování
f_s	–	Mezní frekvence pro nepropustné pásmo
f_p	–	Mezní frekvence pro propustné pásmo
A_p	–	Přípustný pokles magnitudy přenosové funkce v propustném směru
A_s	–	Přípustný pokles magnitudy přenosové funkce v nepropustném směru
α_p	–	Přípustný pokles magnitudy přenosové funkce v propustném směru v dB
α_s	–	Přípustný pokles magnitudy přenosové funkce v nepropustném směru v dB

Obsah

Seznam tabulek

Seznam obrázků

Úvod

Na trhu je mnoho výpočetních prostředí, které se zabývají problematikou zpracování signálů, což je v dnešní době velmi důležitá oblast. Mnoho programů na toto zpracování je ale licencovaných, i proto bych chtěl seznámit čtenáře mé práce s možnou alternativou, kterou je freewarové prostředí SMath Studio, které je na rozdíl od jiných obdobných programů volně stažitelné bez licence. Výhodou SMath Studia je určitě jeho nenáročnost a velmi dobrý výkon při potřebách výpočtů, zobrazování do grafů a celkovým zpracováním signálů.

Cílem této práce je seznámit uživatele s prostředím SMath Studia a ukázat základní práci a možnosti v něm. Zároveň chci ukázat studentům možnost práce v programu, který je volně stažitelný. V první části seznamuji uživatele s prostředím SMath Studia a popisuji jeho instalaci, jak pro operační systém MS Windows, tak ale i pro Linux. Práce obsahuje i popis instalace pomocí názorných obrázků. Ve třetí části mé práce se zabývám prací se třemi druhy proměnných – skalárními, vektorovými a maticemi, každá část práce obsahuje i názorné obrázky příkladů z tohoto prostředí. Čtvrtá část práce je pak zaměřená na práci s 2D a 3D grafy opět s ukázkami. Od páté části se pak moje práce zabývá konkrétními výukovými scripty(tvorba IIR filtrů) z oblasti zpracování signálů. Script obsahuje popis a návod na jeho použití.

V závěru práce je subjektivní srovnání práce v prostředí SMath Studia oproti licencovanému programu Matlab.

1 Seznámení se systémem SMath Studio

SMath Studio je freeware, který je volně stažitelný například z oficiálních stránek SMath Studia, na kterých naleznete také spoustu příkladů řešených v něm. Na oficiálních stránkách je i fórum, kde se uživatelé dělí se svými postřehy a vylepšeními. Smath Studio je dostupné jak pro distribuce operačního systému Windows tak ale i Linux.

Software vytvořil v roce 2005 Andrey Ivashov, ruský matematik ze St. Petersburgu.

SMath Studio je rozsáhlé výkonné matematické studio s rozhraním ve stylu papíru a obrovským množstvím výpočetních funkcí a integrovaným matematickým slovníkem. Umožňuje řešit složité výpočty a sestavovat matematické postupy. SMath si lehce poradí s mocninami, logaritmy, závorkami ve vzorcích a dalšími prvky, které při ručním počítání zaberou spoustu času, a může dojít lehce k chybě. SMath Studio umožňuje výsledky zobrazit do křivkového grafu s možností následného exportu ve formě obrázku (platí pro základní verzi grafů bez pluginu).

2 Instalace SMathStudia

2.1 Stažení SMathStudia

Chci používat SMathStudio a nevím, jak na to? Nejprve je potřeba si SMathStudio stáhnout. Jedná se o volně stažitelný program, který je dostupný například z oficiálních stránek SMathStudia: `http://en.smath.info/`

Po stažení je potřeba přejít k samotné instalaci.

2.2 Instalace pod operačním systémem Windows

1. V prvním kroku se vám zobrazí instalační nabídka Wizard, ve které bude instalace probíhat.
2. Ve druhém kroku je potřeba si přečíst licenční podmínky, a pokud chcete program instalovat, tak s nimi souhlasit.
3. Ve třetím kroku je možnost si vybrat, jestli proběhne instalace pouze pro jednoho uživatele systému (Only for me) nebo je možnost vybrat instalaci pro všechny uživatele (Everybody). A zvolíme možnost další (next).
4. Ve čtvrtém kroku můžeme vybrat, kde se nám vytvoří zástupce. Je možnost vytvořit ho na plochu nebo do START menu. A následně opět zvolit NEXT.
5. Jako další krok je potřeba zvolit cestu, kam budete chtít program instalovat. To zvolíme jednoduše přes tlačítko Browse. A vybereme místo na disku, kde program nainstalujeme. Následně opět zvolíme NEXT.
6. V předchozích krocích jsme nastavili instalaci podle vlastních potřeb a nyní je potřebné samotnou instalaci spustit. To uděláte jednoduše přes tlačítko Install.
7. Nyní už je instalace hotová. V posledním kroku si můžeme vybrat možnost, zdali SMathStudio rovnou spustíme nebo jen dokončíme instalaci. Vše ukončíme a potvrdíme pomocí tlačítka Finish.

2.3 Instalace pod systémem Linux

- Stáhneme z webu SMath studia verzi pro Linux. Opět z výše uvedeného odkazu.
- Stažený soubor (například: SMathStudioDesktop.0975737.Mono.tar.gz) rozbalíme (extrahujeme).
- Rozbalený soubor můžeme přesunout do námi vytvořené složky v systému.

  Lze v grafickém prostředí nebo pomocí příkazové řádky. Zaměříme se na příkazový řádek. Nejprve si ale stáhneme aktualní verze programu mono.

  `sudo mkdir /etc/SMath` (lze zvolit umístění i jinde, dle potřeby)

```
josef@josef-VirtualBox:~$ sudo apt-get install mono-complete
[sudo] password for josef:
Čtu seznamy balíků… Hotovo
Vytváří se strom závislostí
Čtu stavové informace… Hotovo
mono-complete je již nejnovější verze.
0 aktualizováno, 0 nově instalováno, 0 k odstranění a 211 neakt
ualizováno.
```

Obrázek 2.1: Mono stažení

```
josef@josef-VirtualBox:~$ mkdir /etc/SMath
```

Obrázek 2.2: Vytvoření složky pomocí příkazové řádky

Následně soubory přesuneme.

`sudo mv /Stažené/SMathStudioDesktop.0975737.Mono/* /etc/SMath`
(Vždy odkud - kam, opět lze volit dle potřeby. Počítáme s tím, že budete mít soubor stažený ve složce stažené.)

```
josef@josef-VirtualBox:~$ sudo mv ~/Stažené/SMathStudioDesktop.
0_97_5737.Mono/* /etc/SMath
[sudo] password for josef:
josef@josef-VirtualBox:~$
```

Obrázek 2.3: Přesun souborů do složky

- Lze spustit přímo z grafického prostředí pomocí souboru SMathStudioDesktop.exe nebo pomocí příkazu v příkazové řádce:

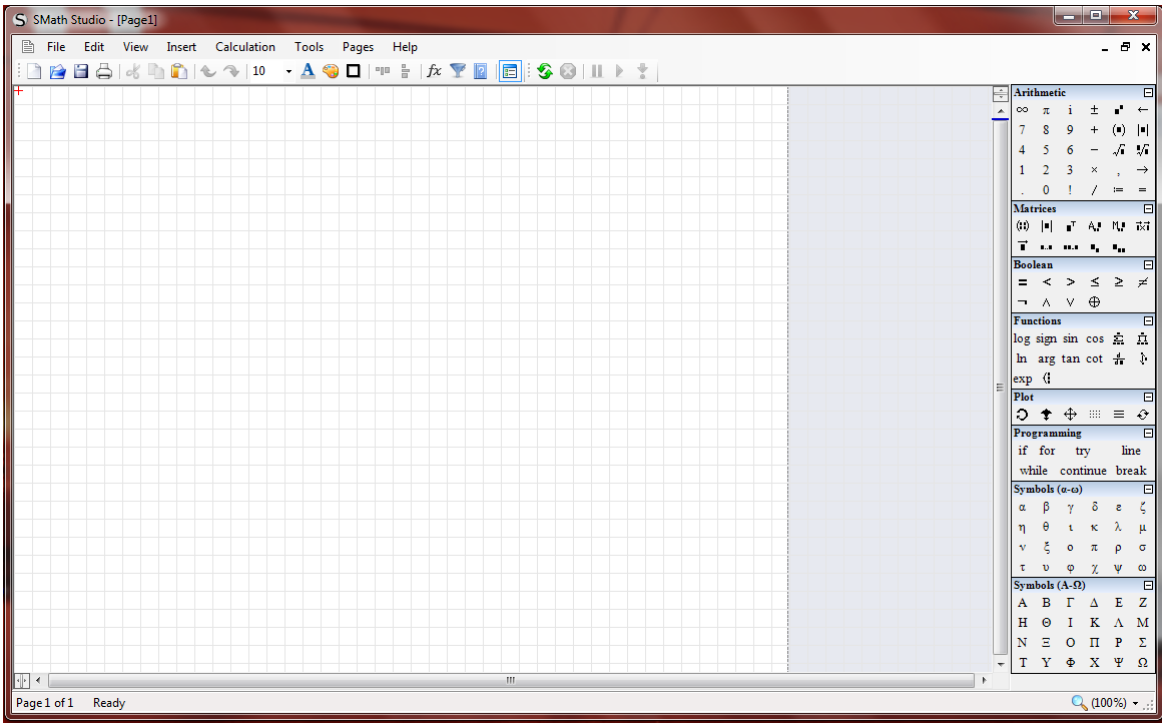
  `sudo mono etc/SMath/SMathStudioDesktop.exe`

```
josef@josef-VirtualBox:~$ sudo mono /etc/SMath/SMathStudio_Desk
top.exe
- [AppDir] HTML Export (1.13.5737.24525)
- [AppDir] Export to image (1.9.5737.24526)
- [AppDir] Picture Region (1.10.5737.24525)
- [AppDir] Plot Region (1.9.5737.24527)
- [AppDir] SMath Studio Files Plugin (1.9.5737.24527)
- [AppDir] Special Functions (1.11.5737.24524)
- [AppDir] Text Region (1.10.5737.24526)
- [AppDir] SMath Viewer Files Plugin (1.1.5737.24528)
- [AppDir] XMCD Files Plugin (1.12.5737.24526)
```

Obrázek 2.4: Spuštění SMath studia

2.4 Seznámení s prostředím

Při spuštění SMath Studia se zobrazí hlavní stránka, která je označená jako Page1, lze změnit dle vašich představ, dle názvu toho, co v prostředí budeme vytvářet. Pracovní plocha je ve stylu čtverečkovaného papíru, po kterém se posouváme a pracujeme na něm. Působí to podobně jako bychom pracovali v sešitě. Čtverečkování se dá vypnout a můžeme pracovat na čistě bílé ploše. Okna s nástroji se dají minimalizovat nebo úplně zavřít. Vše dle potřeby daného uživatele.



Obrázek 2.5: Celkový pohled na pracovní plochu

Na horní liště s nástroji nalezneme spoustu možností, které nám prostředí umožňuje. Pod položkou **File** najdeme možnost, jak založit nový list, upravit daný list, otevřít jiný list pro editaci či uložit daný list. V horní liště následuje položka **Edit**, pokud označíte vybranou proměnnou na pracovní ploše, můžete proměnnou kopírovat, vyjmout či smazat. Pomocí zatrhnutí možnosti show description můžete zobrazit popisek u proměnné. Při zatrhnutí disable evalution můžete proměnnou vyřadit z výpočtů. Pod položkou **View** (zobrazení) se skrývá možnost měnit vzhled SMath Studia. Můžeme odstranit čtverečkování, rozšířit si pracovní plochu i o místa, která by nebyla tisknutelná, ale umožnila by nám větší prostor pro práci. Dále lze zobrazit i výstupní okno.

Pod položkou **Insert** (vložení) se skrývá možnost vkládat matice, funkce (po kliknutí na vložení funkce se otevře okno, které nám umožní z přehledné nabídky vybrat funkci dle představ, pokud je ve SMath studiu implementována), operace, obrázky, grafy, pole s textem, ale i obrázek na pozadí stránky. **Calculation** nabízí možnost nastavení kalkulace

výpočtů na pracovní ploše. Lze nastavit auto - kalkulace, ale také varianty, kdy musíte po každé úpravě nechat stránku přepočítat ručně (zvolit calculation).

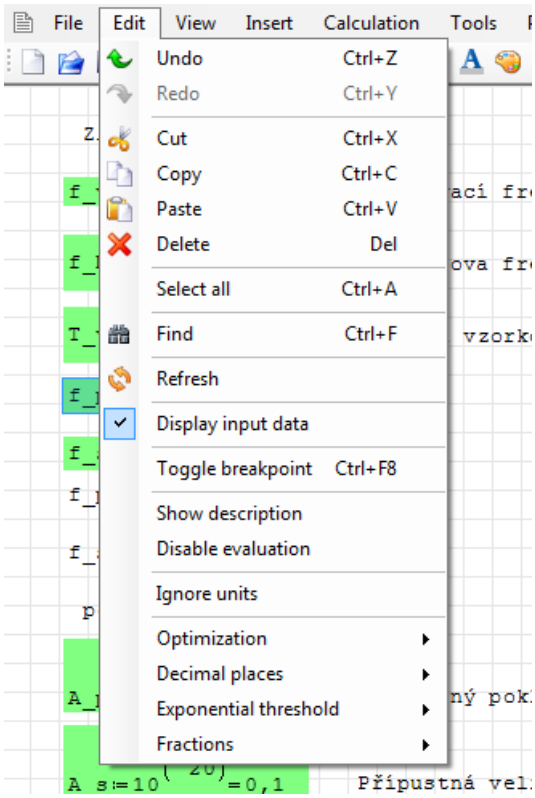
Zajímavou položkou je pak **Tools**, kde můžete naistalovat či vyhledat plugin. Pluginy jsou velkou výhodou SMath studia a suplují chybějící funkce. K pluginu se ještě v průběhu dostaneme. Položka **Pages** nám umožní založit novou stránku (list), můžeme zavřít stránku (list) a vidíme přehled všech otevřených listů. Položka **Help** (nápověda)nám umožňuje možnost velké nápovědy i s příklady.

V pravé části otevřeného programu jsou pak položky pro urychlení práce. Matice, aritmetické operace, boolean prvky, funkce, programovací část, symboly velké a malé řecké abecedy a podobně. Tento celý panel lze pod ikonkou show/hide side panel skrýt nebo nechat zobrazený.

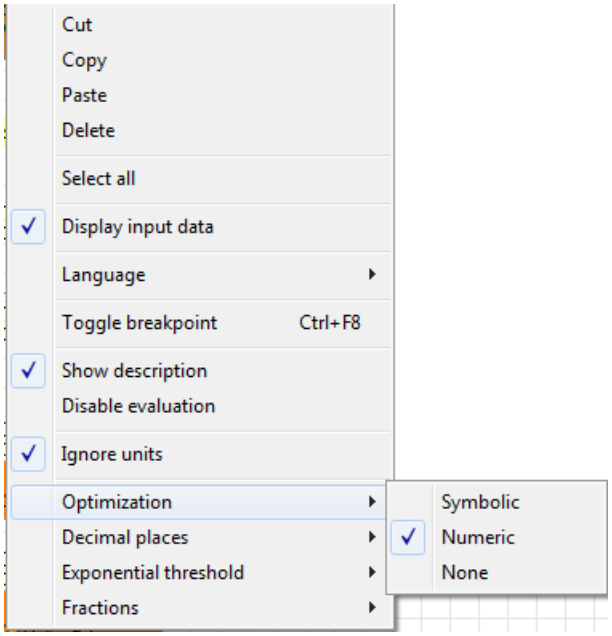
V prostředí se dá pracovat velmi snadno, pokud znáte názvy jednotlivých funkcí, které chcete ve SMath studiu využít. Například pokud chceme vložit do proměnné absolutní hodnotu nějaké jiné proměnné nebo výpočtu, stačí napsat do prostředí čtverečkovaného papíru klíčové slovo abs. Navíc Vám pomůže s přesným výberem funkce našeptávač.

Pokud programujete složitější postup výpočtů a chcete si příklad postupně ladit, lze ve SMath studiu přeskočit proměnnou. Stačí označit tu část výpočtu, kterou nechcete při výpočtech zahrnout, následně v horní liště vybrat položku EDIT a v rozbalovacím menu zvolit možnost DISABLE EVALUATION a do výpočtu se tak proměnná nezahrne. V opačně případě lze stejně proměnnou opět vrátit.

Dalším zajímavým aspektem SMath studia je možnost volit optimalizaci výpočtů. Pokud označíme vybranou proměnnou a klikneme na ni pravým tlačítkem myši, objeví se nám nabídka možností (vyobrazena na obrázku 2.7). V této nabídce zvolíme položku optimalization a následně si můžeme vybrat jestli se bude počítat numericky, symbolicky nebo bez optimalizace.



Obrázek 2.6: Vyřazení proměnné z výpočtů



Obrázek 2.7: Optimalizace

3 Popis práce s proměnnými typu „skalár“, „vektor“ a „matice (pole)“ s ohledem na indexování jednotlivých prvků

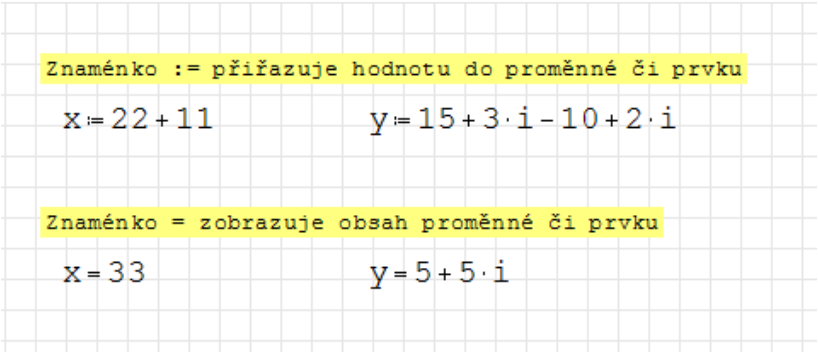
Ve SMath studiu můžeme vytvářet a pracovat s různými druhy proměnných. Můžeme pracovat s jednoduchou skalární veličinou (tedy proměnou, která bude určena jednoduchou číselnou hodnotou, popřípadě reálnou a imaginární složkou), vektory a maticemi (poli).

3.1 Práce s proměnnou typu „skalár“

Proměnná, která bude typu skalár, bude proměnná, která bude určena jedním číselným údajem. Můžeme zde pracovat i s komplexními čísly. Pokud pracujeme s komplexními čísly, můžeme počítat i samostatně s reálnou nebo imaginární složkou. K tomu slouží ve SMath studiu dvě funkce:

- Re(komplexni promenna) - výsledkem bude reálná část komplexního čísla
- Im(komplexni promenna)- výsledkem bude imaginární část komplexního čísla

Do prostředí SMath studia stačí kliknout myší na místo, kde chceme začít kód psát a rovnou tak přejdeme na samotnou tvorbu. Pokud chceme do proměnné přiřadit nějakou hodnotu musíme použít := (tedy symbol přiřazení). Pokud vytváříme proměnnou poprvé tak se po stisknutí rovnítka sám tento přiřazovací znak vytvoří. Pokud už proměnná stejného jména existuje, tak se nám automaticky ukáže její obsah, který si můžeme vypsat na obrazovku na daný list. K tomu nám stačí napsat jednoduše název proměnné a znaménko rovno, za něj se nám automaticky vypíše hodnota proměnné.

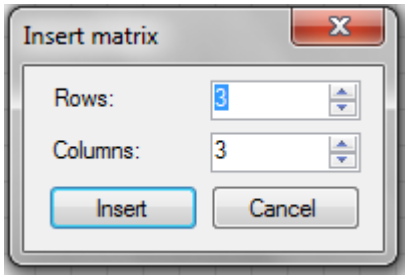


Obrázek 3.1: Práce s reálnou skalární proměnnou a komplexním číslem

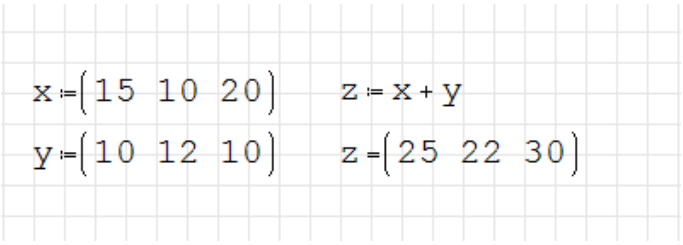
3.2 Práce s proměnnou typu „vektor“

Proměnnou, kterou označujeme jako vektor, můžeme popsat tak, že se jedná o proměnnou, která má svou velikost (stejně jako u skalární veličiny), ale navíc má udán i směr.

Na vektory můžeme pohlížet jednodušeji také jako na matice. Pokud budeme chtít zapsat vektor v SMathu musíme si vytvořit matici o velikosti jednoho řádku a n sloupcích (např. matice $1 \cdot n$ – zde se jedná o tzv. řádkový vektor). Pro zapsání vektoru je potřeba z nabídky Matrices vybrat matici, pokud si na ni kliknu, zobrazí se mi nabídka, která se zeptá, kolik řádků a sloupců chci vytvořit. Zde si vyberu právě zmiňovanou možnost. Pro zrychlení práce, ale nemusíme nikde klikat a stačí použít klávesovou zkratku Ctrl + M, následně vyskočí opět stejná nabídka.



Obrázek 3.2: Nabídka velikosti matice

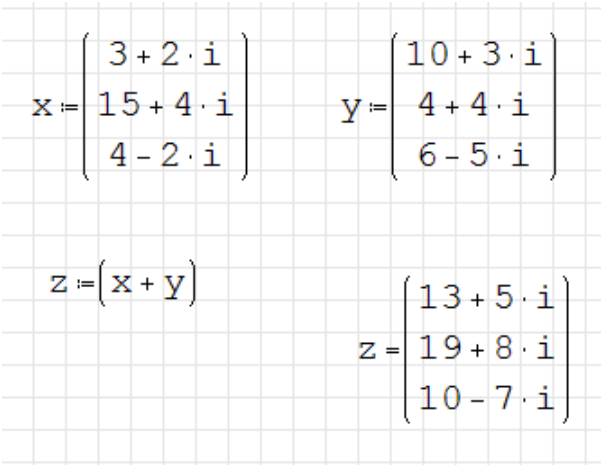


Obrázek 3.3: Řádkové vektory a práce s nimi

Pochopitelně je zde možnost vytvořit si i sloupcový vektor. U něj budeme postupovat stejně, pouze se jedná o matici velikosti n řádků a jednoho sloupce $(n \cdot 1)$.

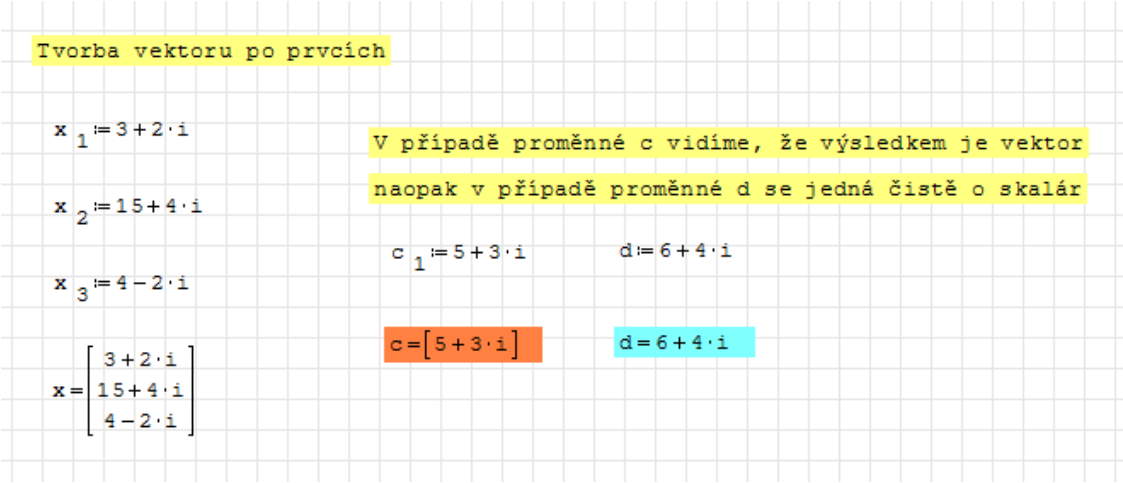
Práce s takovými vytvořenými vektory je pak jednoduchá a ve své podstatě se zde pracuje stejně jako u skalární proměnné. S tím rozdílem, že výsledkem například součtu dvou matic je matice, kde se sčítá složka x_1 se složkou y_1, x_2 s y_2 a x_3 s y_3. Opět je zde možnost pracovat i s komplexními čísly. Z hlediska indexování se pak koukáme na vektor jako na pozice, které vyžadují pouze jediný index. (Stačí nám znát pouze pozici v řádku (u řádkového vektoru) nebo ve sloupci (u sloupcového vektoru). Například vektor x je složen z $x_1 = 3 + 2i, x_2 = 15 + 4i, x_3 = 4 - 2i$. Jednoduše řečeno na rozdíl od matic nám stačí udat pozici, kde se prvek vektoru nachází v rámci jednoho údaje (u matic pak potřebujeme znát pozici na řádku i ve sloupci).

Tak jako máme možnost nechat si vypsat celý vektor, který chceme, tak je i možnost vypsat si pouze hodnotu na daném indexu. U vektoru stačí zadat název matice a jako spodní index zadáme číslo řádku nebo sloupce, za rovnítko se nám vypíše hodnota na dané pozici. Ke správnému zapsání jednotlivých prvků je potřeba využít funkci el (anglicky elements). Vytvoří nám v pracovním prostředí spodní index, do kterého můžeme



Obrázek 3.4: Ukázka práce se sloupcovým vektorem s komplexními čísly

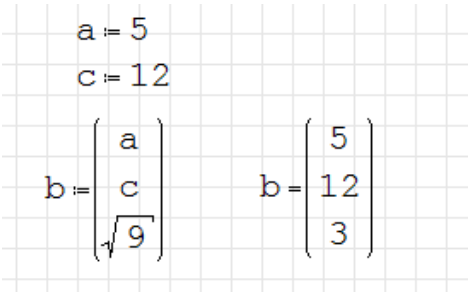
napsat námi požadovaný číselný údaj, tedy pozici prvku ve vektoru. Zároveň, jak vidíme na obrázku 3.5 můžeme vektor vytvořit i například vkládáním jednotlivých prvků na dané pozice. Zde je potřeba si všimnout, že pokud vkládáme na nějakou pozici námi zadanou hodnotu, vytvoří se nám vektor. Pokud bysme indexaci nepoužili, vytvoří se nám skalární veličina.



Obrázek 3.5: Indexace vektoru

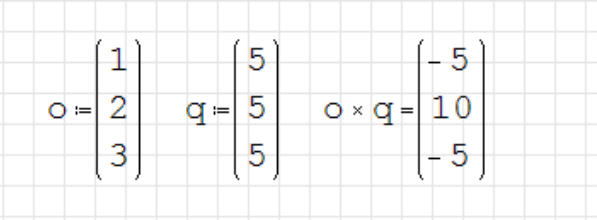
Jak vidíme na obrázku3.6 vektor můžeme vytvořit nejen zadáním veličiny, ale také jiné proměnné nebo přímo výpočtem na dané pozici. To nám umožňuje vytvářet vektory v závislosti na výsledcích jiných výpočtů.

Vektory můžeme násobit, sčítat, dělit, ale i odečítat jako matice. Je zde i možnost vypočítat si vektorový součin dvou vektorů. K tomu nám slouží funkce Cross product v nabídce Matrices. (Nalezneme jej označený jako x s šipkou nad písmenem).



Obrázek 3.6: Zápis vektoru proměnnou nebo výpočtem

Na obrázku 3.7 si ukážeme příklad se dvěma vektory (o, q).



Obrázek 3.7: Vektorový součin

Funkce range nám umožní tvořit vektory v určitém rozsahu a po daném kroku. Funkce range má opět dvě varianty. Máme variantu range(2), zde se dá určit jen, z jakého rozsahu prvky budou, automaticky s krokem 1. Pokud zvolíme variantu range(3), tak si můžeme určit i daný krok. Ve SMath studiu nám funkce range vytvoří vždy sloupcový vektor. Je tedy velmi vhodné v tomto prostředí pracovat především se sloupcovými vektory.

- range(2) –> 0 .. 10 (např. z intervalu 0 – 10, krok je automaticky dán na hodnotu 1)
- range(3) –> 0; 0,1 .. 10 (např. z intervalu 0 – 10 s krokem 0,1)

3.2.1 Funkce stack, augment

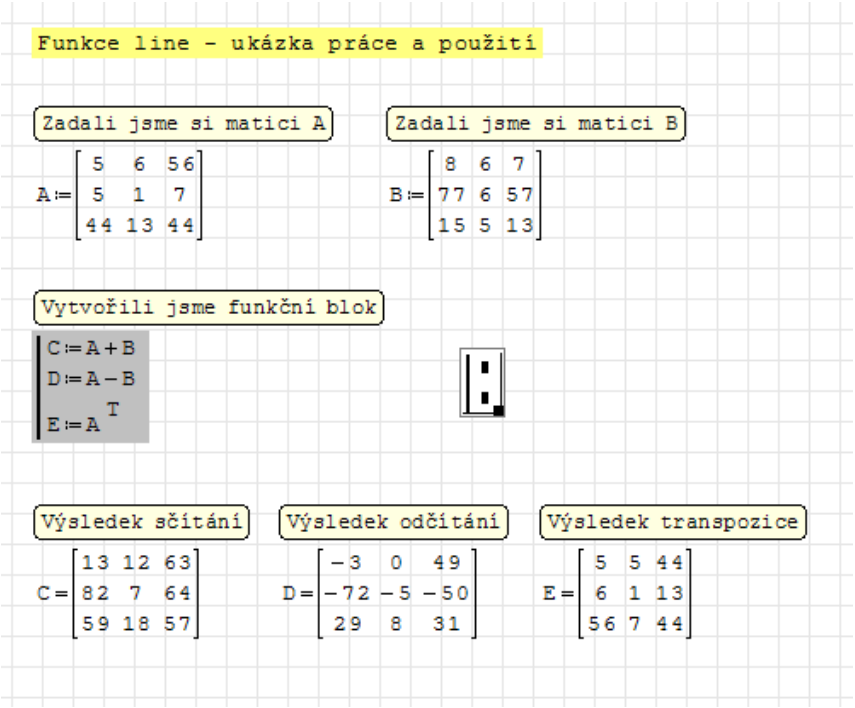
SMath studio nabízí dvě zajímavé funkce, které si ukážeme na praktické úloze z oblasti zpracování signálů v praktické části práce.

- Stack() - tato funkce vezme vektory uvedené v závorce (oddělujeme je čárkou) a vloží je v pořadí jaké je uvedené v závorce za sebe. Tzn. dva sloupcové vektory vytvoří jeden sloupcový vektor, kde jsou nejprve řazeny jednotlivé prvky prvního vektoru, pak druhého vektoru až do prvků posledního vektoru uvedeného v závorce.
- Augment() - tato funkce vezme vektory uvedené v závorce (oddělujeme je čárkou) a vloží je vedle sebe v pořadí jaké je uvedeno v závorce. Tzn. pokud máme dva

sloupcové vektory, tak se nám z jednotlivých prvků vytvoří dvojice. Pro správnou funkci je ale potřeba zajistit, aby byly oba vektory stejné dimenze. Pokud by nebyly - SMath studio nás upozorní chybovou hláškou *"Array dimensions do not match"*.

3.2.2 Funkce line na tvorbu funkčních bloků

Ve SMath studiu můžeme přehlednost kódu zajistit i využitím funkčních bloků 3.8, které nám sjednocují určitou skupinu výpočtů. Výhodou je dobré odlišení celého bloku určitou barvou, případně pokud bychom chtěli všechny výpočty vyřadit, stačí nám označit celý blok a zvolit disable evalution. Nevýhodou ale je, že si výsledky jednotlivých kroků můžeme vypsat až po skončení celého bloku, tj. také nevýhoda i u použití cyklů, kdy si nemůžeme jednotlivé kroky zobrazit. Defaultně je nastavena funkce line (stačí do pracovní plochy napsat klíčové slovo funkce line) na dva prvky. Můžeme ji ale prodloužit a nastavit si počet řádků (prvků) dle vlastní potřeby. Stačí klinout na černou svislou čáru a v pravém dolním rohu se objeví černý čtvereček za který, když zatáhneme levým tlačítkem myši se začnou přidávat řádky.

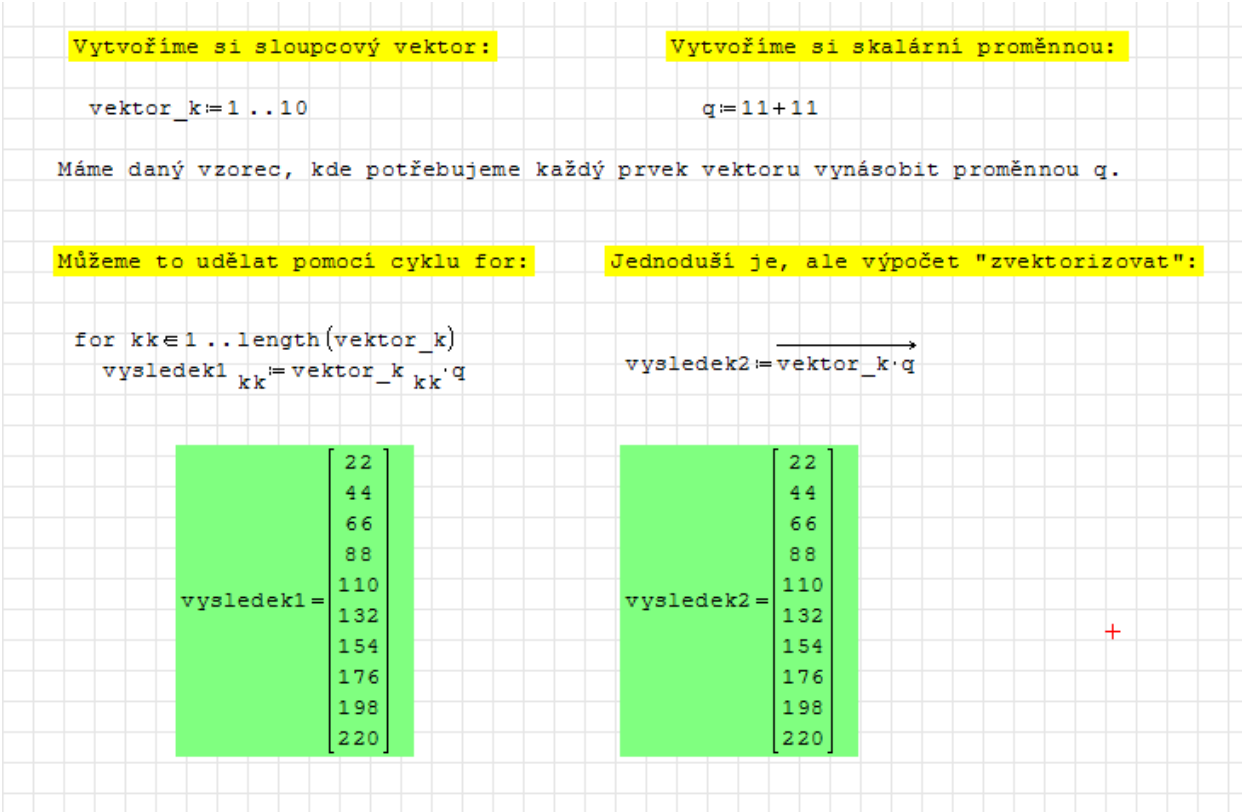


Obrázek 3.8: Ukázka funkce line

3.2.3 Vektorizace výpočtů, práce s funkcí Vectorize

Pokud bychom chtěli pracovat s jednotlivými prvky vektoru, můžeme je procházet klasicky cyklem for a jednotlivé prvky vektoru například násobit s jinou proměnou. To ale

není ve SMath studiu nutné. Vše se dá vyřešit pomocí funkce Vectorize function (v pravém menu SMath studia, je vyobrazena jako šipka nad černým čtverečkem). Jednotlivé výpočty se tak "zvektorizují"a prostředí už dokáže ve výpočtu poznat, že se jedná o vektor s určitým počtem prvků. Vše je ukázáno na obrázku 3.9



Obrázek 3.9: Ukázka vektorizace výpočtu

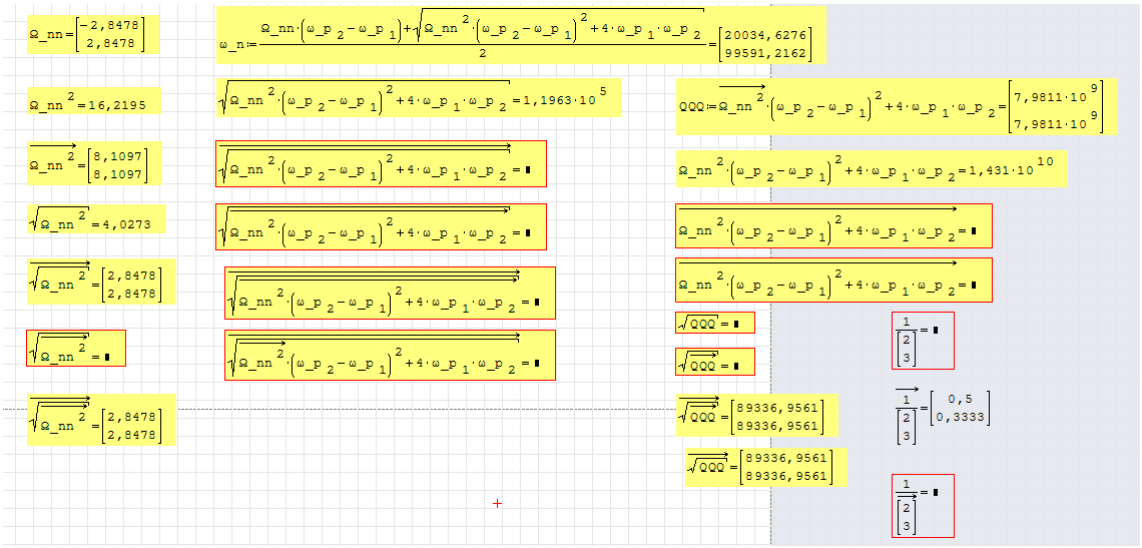
V případě, že si chceme vypsat určitý prvek vektoru na dané pozici, musíme použít u proměnné spodní index. Ve SMath studiu je to u vektoru nazváno Vector element. Máme tři možnosti, jak tento index na pracovní plochu vložit.

1. Vybereme si ho z nabídky v pravém sloupci prostředí v sekci Matrices
2. Vložíme ho pomocí klávesnice, a to když napíšeme do pracovní plochy: ([)
3. Napíšeme do pracovní plochy el a z nabídky našeptávače vybereme el(2)

Ve SMath studiu jsem v průběhu tvorby bakalářské práce objevil funkci Vectorize. Jedná se o zajímavou funkci, kterou jsem důkladně prozkoumal. SMath studio dokáže překvapit, že pokud Vám něco funguje na jednom řádku, nemusí Vám fungovat to stejné na řádku dalším.

Funkce Vectorize umožňuje pracovat s proměnnými typu vektor a matice bez nutnosti použití cyklů. Narazil jsem, ale na několik zajímavých skutečností.

Na obrázku 3.10 je zobrazeno několik možností a různých zápisů. V případě, že jsem chtěl zvektorovat celou odmocninu, pod níž se nacházela proměnná typu vektor a další výpočty, tak SMath studio vyhodilo chybovou hlášku: *„Requested matrix element doesn't exist"*. Jak už jsem, ale naznačil, pokud jsem tuto stejnou variantu aplikoval na celou odmocninu, pod níž byla pouze proměnná typu vektor, tak SMath studio dokázalo správně zobrazit výsledek.



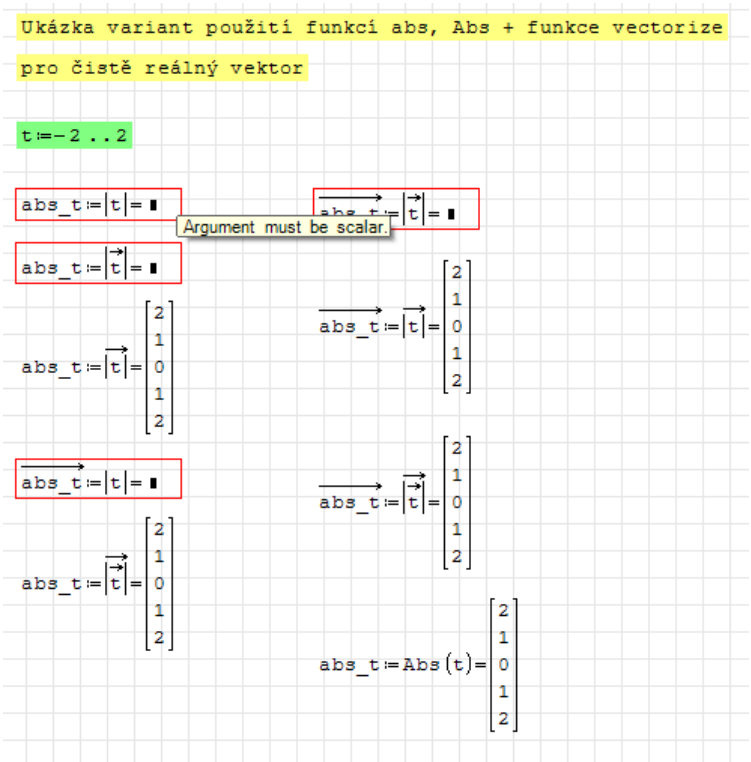
Obrázek 3.10: Ukázka možností vektorování

V návaznosti na tuto zajímavou skutečnost jsem zkusil výpočty pod odmocninou vložit do pomocné proměnné. Následně jsem zvektoroval odmocninu, pod kterou byla pomocná proměnná – výsledkem byl správný výpočet. V případě IIR filtrů jsem takto chtěl funkci aplikovat a snížit tak nutnost cyklů. Na jednom řádku bylo vše pořádku, na druhém u analogicky stejného vzorce už ne. Byl jsem tedy nucen použít cykly. Z toho tedy plyne, že v případě SMath studia používejte spíše cykly, jelikož například u složitějších výpočtů s odmocninami nemusí vše fungovat dle Vašich představ.

Dalším zajímavým faktem je správné použití funkcí abs a Abs. Funkce abs slouží pro proměnné typu skalár. Jsou pro ně přednostně určeny a v těchto případech pracuje funkce správně. Funkce Abs je přednostně určena pro proměnné typu vektor a matice. V této souvislosti jsem otestoval různé možnosti použití na praktickém příkladu. Vše navíc i s ohledem na funkci vectorize.

Na obrázku 3.11 vidíme použití těchto funkcí na proměnnou, která typu vektor a má čistě reálné složky. Pokud použijeme na vektor funkci abs, tak musíme výpočet pomocí funkce vectorize správně zvektorovat. Zde existuje několik možností a kombinací, jak můžeme výpočty vektorovat. V případě, že nezvektorujeme výpočet vůbec, SMath studio nás upozorní jako v ostatních špatných případech chybovou hláškou *"Argument*

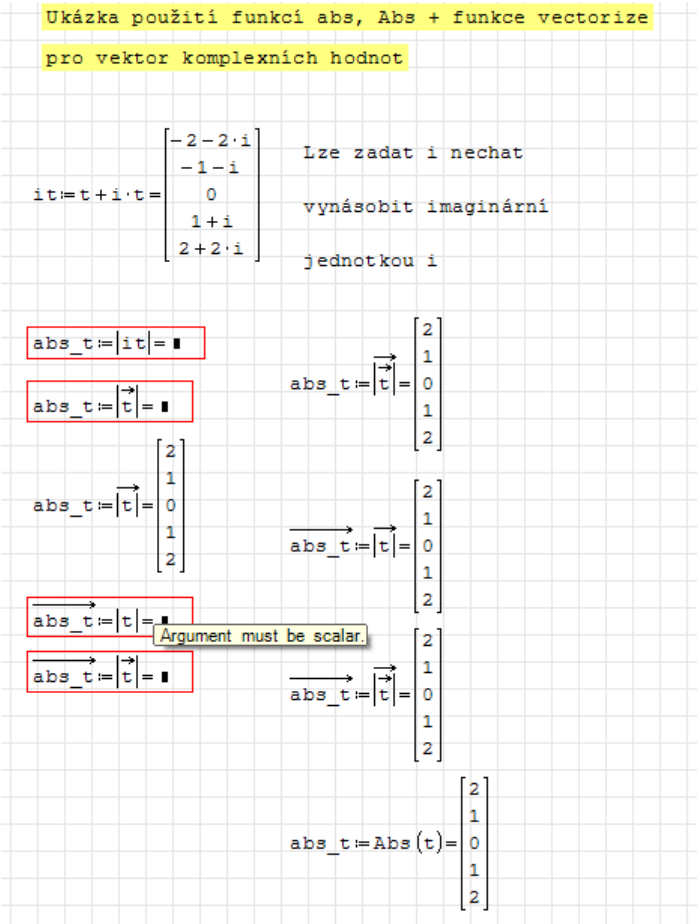
must be scalar." V případě, že použijeme funkci Abs nemusíme do výpočtu vektorování vůbec vkládat, jelikož, jak už bylo zmíněno, je tato funkce určena právě pro vektory. U funkce abs nefunguje, pokud zvektorujete pouze proměnnou uvnitř absolutní hodnoty ani pokud zároveň zvektorujete proměnnou, do které se má výsledek uložit. Stejně tak nefunguje ani zvektorování pouhé proměnné, do které budeme ukádat. Ostatní varianty (u některých případů i dvojité vektorování) fungují. Můžeme tak při správném použití funkce vectorize použí pro výpočet obsaolutní hodnoty vektoru i funkci abs.



Obrázek 3.11: Ukázka funkcí vectorize, abs a Abs na skalární proměnné

Stejnou analýzu jsem provedl i s vektorem, který má reálnou a imaginární složku (na obrázku 3.12), tedy jeho hodnoty jsou komplexní čísla. I zde fungují všechny kombinace stejným způsobem jako v případě, že je vektor čistě reálný.

Zajímavou skutečnost jsem objevil u vektorování výpočtů v případě použití funkce tan. Na obrázku 3.13 vidíme, že pokud jsou vstupní proměnné vektory, které mají pouze jeden jediný prvek a my zvektorujeme právě výpočet s těmito vektory, tak nám SMath studio zahlásí chybu *"Arguments must be scalar"*. Na rozdíl od ukázky 3.14, kde vidíme, že vstupní vektory jsou dimenze 2 (obsahují dva prvky), tak zvektorování proběhne vpořádku. Při tvorbě ve SMath studiu je potřeba si tyto skutečnosti hlídat a mnohdy nás mohou nemile překvapit. Řešením této situace je použití cyklu, který proběhne korektně ve všech případech a je jedno jaké dimenze vstupní vektor bude. Při tvorbě v praktické části této práce jsem musel několikrát použít cyklus namísto funkce vectorize. Ve SMath



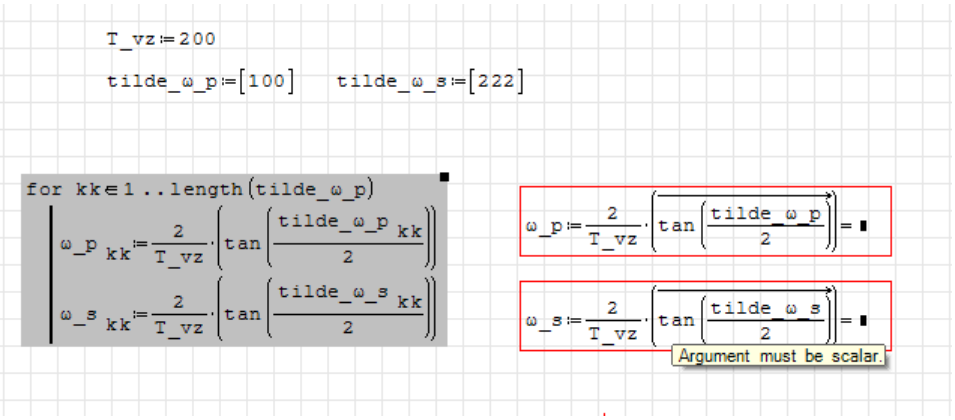
Obrázek 3.12: Ukázka funkcí vectorize, abs a Abs na komplexní proměnné

studiu je práce s touto funkcí často problematická a ukazuje na jeden z nedostatků tohoto programu.

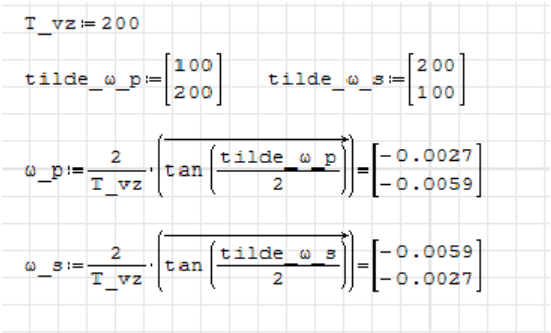
3.3 Práce s proměnnou typu „matice (pole)“

Ve SMath studiu je možnost tvořit matice několika způsoby. Mimo již uvedené, je možné tvořit matici pomocí funkce, kterou stačí napsat do prostředí matrix(počet sloupců, počet řádků).

Pokud chceme dále s jednotlivými prvky matice pracovat, je potřeba využít funkce el (anglicky elements). Zde je možnost si jednotlivý prvek matice vypsat, zapsat do něj, zkrátka dále pracovat. Je možnost si vybrat ze dvou variant el(2) a el(3) liší se v počtu indexů. U el(2) si vypíšeme jednodimenzionální prvek z matice u el(3) dvojdimenzionální, zde už je varianta si vypsat prvek x-tého řádku v y sloupci.



Obrázek 3.13: Ukázka vektorování funkce tan - jednoprvkový vektor



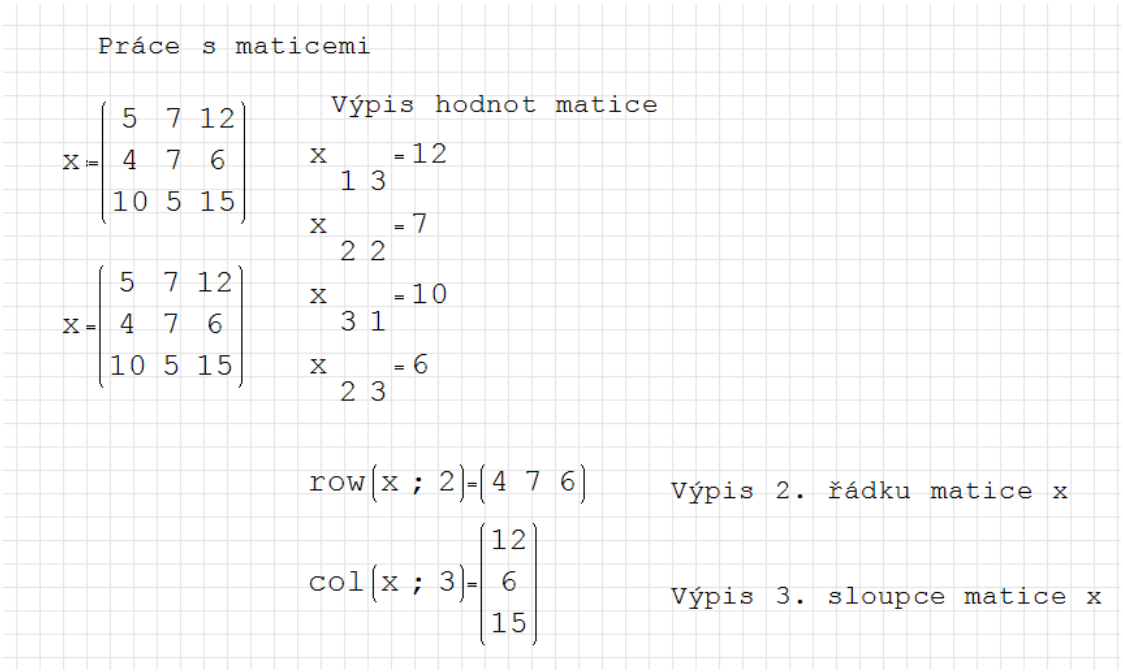
Obrázek 3.14: Ukázka vektorování funkce tan - dvouprvkový vektor

Pokud byste nevyužívali funkci el, nebude Vám výpis ani zápis jednotlivých prvků matice fungovat. SMath studio pak bere název proměnné například x_{12} jako proměnnou s tímto názvem a dolní index se nebude aplikovat.

Proměnná typu matice (pole) se od vektoru liší počtem svých řádků a sloupců. Můžeme sice vytvořit matici o 1 sloupci a x řádcích, ale zde se jedná především o větší pole. Obecně můžeme vytvořit matici o n sloupcích a n řádcích. Opět k tomu použijeme klávesovou zkratku Ctrl + M. K indexaci se musí použít dvě hodnoty, použijeme dva indexy (index řádku a index sloupce). Například pro matici A o 3 řádcích a 3 sloupcích by výpis hodnot na jednotlivých pozicích vypadal následovně: $A_{11} = hodnota$ $A_{21} = hodnota$ $A_{31} = hodnota$ (takto jsme si vypsali celý první sloupce). Dále by se pokračovalo analogicky.

Můžeme si také nechat zobrazit celý jednotlivý řádek nebo sloupce najednou. K tomu nám bude sloužit jednoduchý zápis:

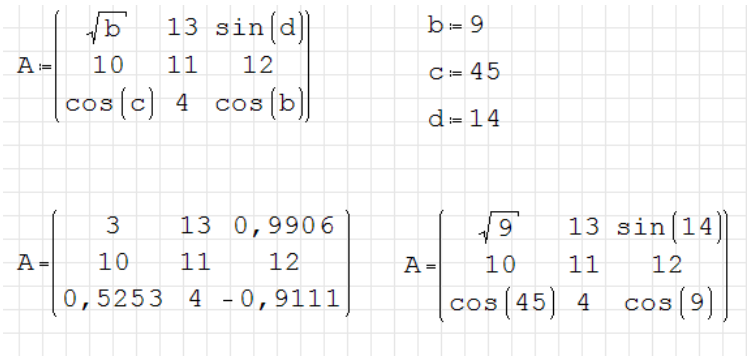
- row (nazev matice, cislo radku)
- col (nazev matice, cislo sloupce)



Obrázek 3.15: Ukázka indexace matice 3x3 a její výpis

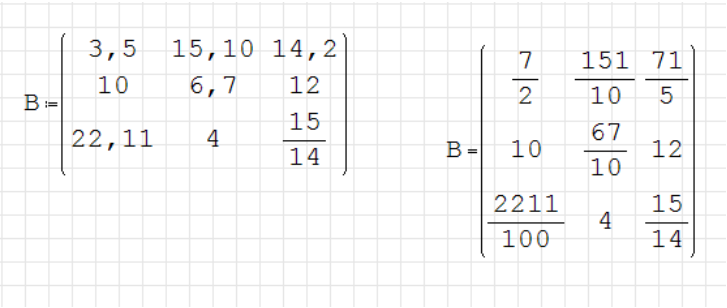
Stejně jako u vektoru můžeme na jednotlivé pozice vložit výsledek jiné proměnné, nějaký výpočet. Máme zde možnost zobrazit si danou matici jak symbolicky, tak ale i číselně. Na obrázku 3.16 můžeme vidět zapsání numericky a symbolicky. Pokud chceme vypsat hodnoty jednotlivých pozic v matici symbolicky, musíme použít místo znaménka rovná se znaménko šipky za název matice dáme šipku (klávesové zkratka Ctrl + .).

Pokud si vypíšeme matici symbolicky, dané výpočty se v matici ještě neprovedou, ale pouze se nám zobrazí dosazení proměnných do výpočtů. Pokud si zobrazíme matici číselně, tak se nám hodnoty na jednotlivých pozicích zobrazí po výpočtu.



Obrázek 3.16: Matice vypsaná numericky x matice vypsaná symbolicky

Na obrázku 3.16 vidíme nahoře vlevo vytvoření matice, nahoře vpravo pak hodnoty jednotlivých proměnných. Dole vlevo vidíme matici vypsanou numericky, kde se nám zobrazí přímo hodnoty po výpočtech. Dole vpravo pak matice vypsaná symbolicky za použití zkratky (Ctrl + .), hodnoty proměnných se pouze dosadili do výpočtů.



Obrázek 3.17: Zobrazení matice numericky se zlomky místo desetinných čísel

Z obrázku 3.17můžeme vidět, že pokud vytvoříme matici, kde se budou vyskytovat desetinná čísla ve tvaru s desetinnou čárkou, tak se nám ve zobrazení matice numericky, převedou dané desetinné čísla na zlomky jim odpovídající. Vždy záleží zda použijeme pro výpis proměnné rovnítko nebo šipku. Například $15,10$ bude zobrazeno jako $\frac{151}{10}$. V maticích můžeme opět používat i komplexní čísla a stejně s nimi i pracovat.

Matice dále mohou obsahovat také více datových typů. Mohou v sobě zahrnovat další matice i více datových typů. Matici, která bude vytvořena z několika matic, vytvoříme zcela jednoduchým způsobem. Na každou pozici v matici vložíme další matici, tím docílíme většího počtu matic v jedné.

Můžeme sestavit i matici, kde budou na prvním řádku číselné hodnoty, na druhém řádku bude na jednotlivých pozicích vypsán text. Dále je možnost ve SMath studiu vytvářet matice naplněné nulami a je zde možnost s maticemi různě pracovat.

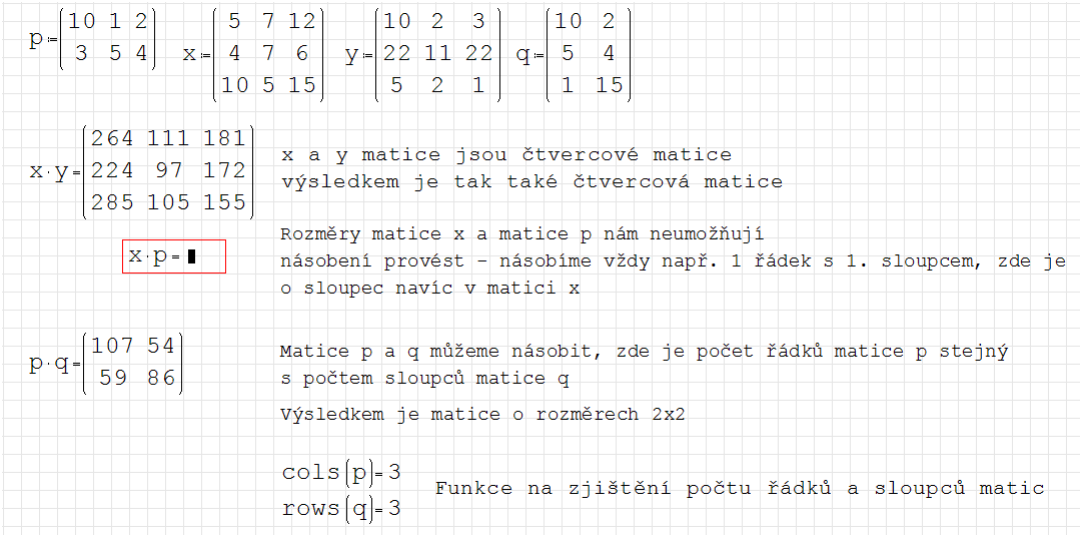
- **Násobení matic** – pokud chceme násobit matice, musíme dodržet podmínku pro násobení matic, které platí v každém matematickém prostředí. Násobené matice musí být buď čtvercové, což je speciální příklad matice nebo musí mít první matice stejný počet řádků jako má druhá matice počet sloupců. Pokud tomu tak nebude, upozorní nás studio chybovou hláškou.

- **Sčítání matic** – u sčítání matic se vždy sečtou prvky na stejné pozici v první a druhé matici.

- **Odečítání matic** – stejné pravidla jako u sčítání

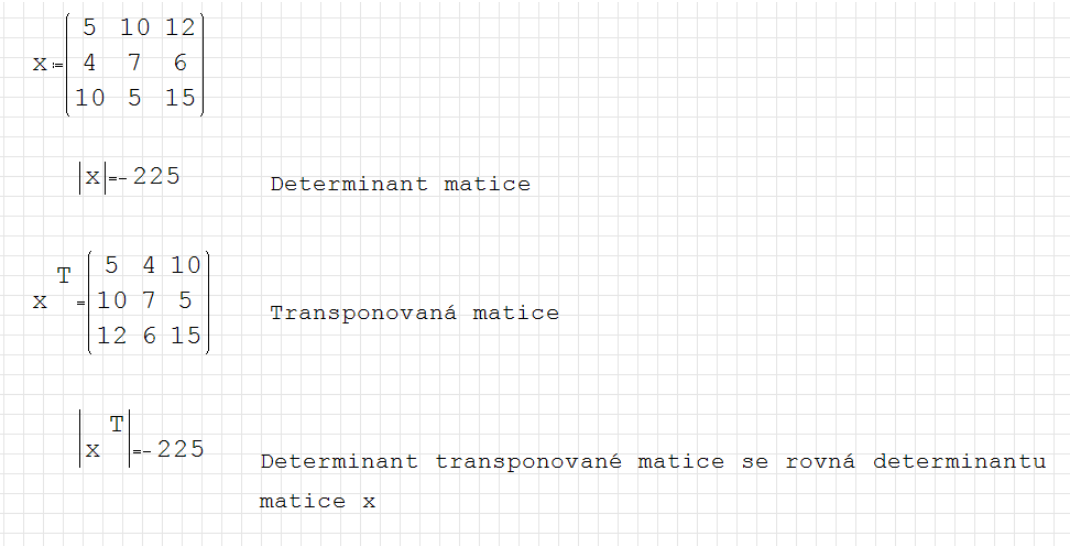
- **Dělení matic** – neexistující operace, u matic je nahrazena inverzní maticí.

- **Determinant matice** – z matice si můžeme nechat vypočítat i determinant. Je potřeba zadat matici. Pokud bychom zadali skalár, pochopitelně by nešla operace provést. Opět se zobrazí chybová hláška upozorňující na tuto skutečnost. Matice musí být čtvercová.



Obrázek 3.18: Násobení matic

- **Transpozice matice** – ve SMath studiu je možné udělat transpozici zadané matice, což znamená, že se nám řádky zamění za sloupce a opačně.



Obrázek 3.19: Determinant a transpozice matice

SMath nám nabízí spoustu připravených funkcí pro práci s maticemi. Funkce můžeme najít v nabídce Insert -> Functions. Zde jsou jednotlivé funkce roztříděny do kategorií podle toho, s čím pracují. Zaměříme se zde na funkce pracující s maticemi (tedy Matrix and vector). Ukážeme si některé z nich.

- Col (matice, cislo sloupce), Row (matice, cislo radku) – vstupem funkce je vybraná matice a číslo sloupce, který chceme vypsat. Výstupem z funkce je pak daný sloupce, který si vybereme.
- Cols (matice nebo vektor), Rows (matice nebo vektor) – vstupem funkce je matice nebo vektor, výstupem je pak počet sloupců v dané matici.
- Det (matice) – vstupem je matice, výstupem je její determinant.
- Diag (cislo popripade matice) – vstupem je číslo nebo matice, výstupem je pak matice, která bude mít na diagonále zadané hodnoty dle dimenze, kterou vybereme. Pokud vložíme matici o 3 řádcích a 1 sloupci, budou hodnoty této matice na diagonále výsledné matice.
- Identity (cislo n) – vstupem je číslo n, které symbolizuje výslednou matici $n\ x\ n$, kde jsou na diagonále jedničky.
- Length (matice nebo vektor) – vstupem je matice nebo vektor, výstupem je pak číslo určující počet prvků v matici.
- Max (matice), Min (matice) – vstupem je matice, výstupem je pak maximální / minimální hodnota v matici.
- Tr (matice) – vstupem je matice, výstupem je pak součet prvků na diagonále matice.
- vectorize (vyraz – vektor, matice nebo system) – umožnuje provést stejnou operaci na každý prvek vektoru, matice nebo systému.

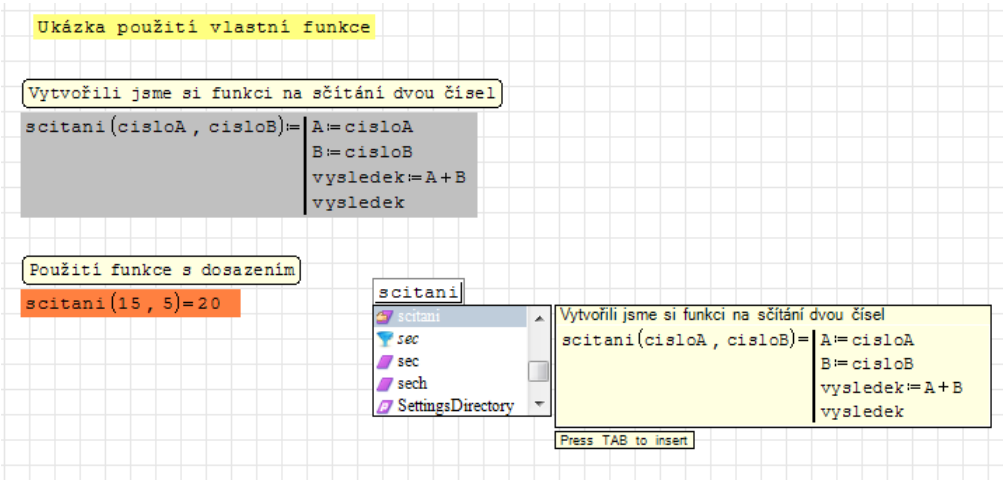
4 Tvorba vlastních funkcí

SMath studio nabízí celou řadu dalších možností a postupů, jak pracovat na určité problematice. Tento matematicko-fyzikální program nezahrnuje některé funkce oproti Matlabu, ale na druhou stranu je v tomto prostředí možno tvořit i vlastní funkce.

Funkce se tvoří poměrně intuitivně. Využívá se k tomu příkaz line, který umožní přiřadit k názvu a vstupním proměnným odpovídající postup s výstupem na konci.

Nazev_funkce(vstupni_promenne):=(line) a odpovídající postup s názvem proměnné na konci, která slouží jako výstup z funkce. Takto vytvořenou funkci pak můžeme, kdekoliv využít. Název funkce se nám vloží do nápovědy při hledání vhodné funkce, proměnné apod., kde se ukáže i konstrukce dané funkce.

Na názornou ukázku se můžete podívat na obrázku 4.1, kde jsem vytvořil triviální funkci na sčítání dvou čísel. Tato funkce by byla ve SMath studiu zbytečná, jelikož sčítání je jednoduchou operací, ale pro názornost stačí. Vytvoříme si odpovídající funkci (v našem případě funkci scitani ~()), za příkazem line si definujeme postup, který se s danými prvky (vstupními proměnnými) bude dít. Na konci je pak potřeba napsat proměnnou, která má být výstupem dané funkce, v našem případě výsledek. Pak už stačí jen do pracovní plochy napsat název funkce a našeptávač nám nabídne tuto funkci, kde zároveň vidíme, jak je funkce řešená (ve žlutém rámečku), pak už stačí jen funkci klikuntím vložit a dopsat vstupní proměnné a můžeme si jednoduše vypsat výsledek funkce.



Obrázek 4.1: Ukázka tvorby vlastní funkce

5 Popis možností grafického zobrazení výsledků výpočtů. 2D a 3D grafy

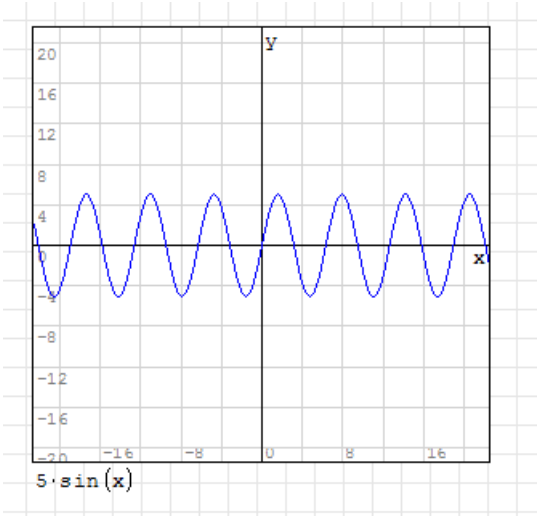
V rámci SMath studia můžeme zobrazovat výsledky do grafů. Máme v něm dva druhy grafů – 2D grafy a 3D grafy. Jednotlivých typů grafů pak může být více. Na fóru SMath studia se dá stáhnout plugin X-Y graph, který nám tak umožňuje ještě efektivněji pracovat s grafickým znázorněním výsledků. Pomocí tohoto typu grafu můžeme měnit popisky u os, znázorňovat graf v bodech nebo ve spojitém zobrazení, nastavovat rozsah na osách a další.

Tento plugin můžeme stáhnout z oficiálního fóra SMath studia:
`http://en.smath.info/forum/yaf_poststst1774_X-Y-Plot-Region.aspx`

Graf vytvoříme a vložíme jednoduše. Stačí si v menu programu vybrat položku Insert -> Plot -> následně vybereme 2D nebo 3D graf. 2D graf můžeme, ale vložit, ale i pomocí stisknutí zavináče @ (například na české klávesnici ALT GR + V). Opět nám to může zjednodušit a zrychlit práci se vkládáním grafu. U varianty X-Y graph musíme použít variantu vložení přes Insert->X-Y Plot. V následující kapitole si ukážeme typ jednoduchého typu grafu, který je součásti SMath studia (X-Y grafu se věnuji v praktické části mé práce), kde jsem se rozhodl využívat možnost pluginu X - Y graph, který je efektivnější a mnohem lépe se s ním pracuje.

5.1 2D grafy

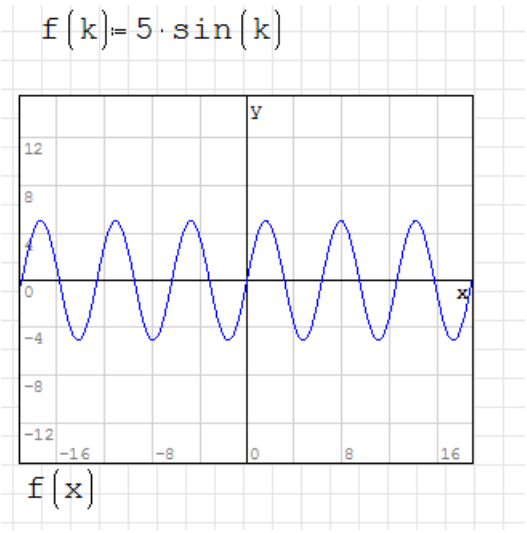
Pro tvorbu 2D grafu potřebujeme minimálně dvě hodnoty. Hodnotu na ose x a hodnotu na ose y. Jednoduše řečeno pro hodnotu x si například pomocí funkcí dopočítáme hodnotu y. Potřebujeme tedy dvojice hodnot.



Obrázek 5.1: Příklad jednoduchého 2D grafu

Na obrázku 5.1 vidíme takový příklad 2D grafu. Jakmile vložíme do listu prázdný graf, musíme ve spodní části pod mřížkou a osami určit podle čeho se bude graf počítat. Výsledkem je pak vytvoření 2D grafu, kde se pro každou x hodnotu dopočítá pomocí funkce $sin(y)$ y hodnota a bod se vykreslí do grafu.

Graf, ale můžeme nadefinovat i funkcí (například funkcí f), která bude mimo okno grafu. Nadefinujeme si například funkci, která bude počítat hodnoty sinu na základě jakéhokoliv vstupu, což nám bude umožňovat použít funkci přímo v okně grafu jako pouhý „odkaz“ na už vytvořenou funkci mimo okno grafu. Pro příklad se podívejme na stejnou funkci jako v předchozím příkladě, jen ji nadefinujeme mimo okno grafu. Hodnoty x se tedy budou brát z grafu a budou vstupovat do naší funkce. (Pro hodnotu x se dopočítá hodnota y z obecného vztahu $f\ (k)\ :=\ 5 \cdot sin(k)$).



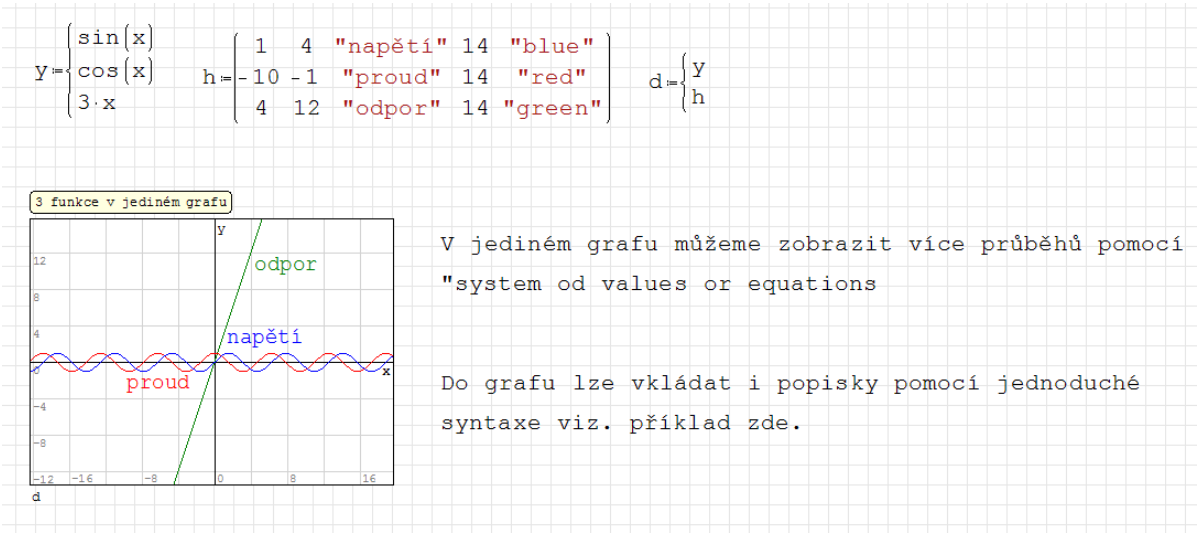
Obrázek 5.2: Ukázka 2D grafu definovaného funkcí mimo okno grafu

Do jednoho 2D grafu si můžeme zobrazit i několik různých průběhů funkcí. Do jednoho grafu můžeme tedy umístit průběh více funkcí, které budou od sebe barevně odlišeny. Takto se nám barevně odliší 6 funkcí v jednom grafu, další funkce už budou následně barvy už jen znovu opakovat. Celkově můžeme do jedné proměnné uložit i sadu funkcí. Provedeme to jednoduše, pokud vybereme v poli Functions na pravé straně vedle pracovní plochy Systém of values or equations (symbol složené závorky).

Standardně se objeví možnost zapsat dvě takovéto to například funkce, ale natažením myši za pravý dolní roh, můžeme přidávat další. Jedná se o vytvoření systému hodnot nebo rovnic, jak vyplívá z českého překladu.

Do grafu můžeme také vkládat popisky jednotlivých průběhů, což můžeme udělat podle vytvořené syntaxe. V našem případě máme nadefinovanou pozici na ose x, pozici na ose y, chars (zde je možnost zadat, co se na daných souřadnicích zobrazí, v našem případě to bude text proud a napětí, ale může zde být vytvořený také jednoduchý bod například pomocí písmenka X, dále velikost daného znaku (u nás velikost písma slova)

a barvu daného bodu (slova). Barvu můžeme zadat jako název barvy, ale také jako hexadecimální číslo. Položky velikost a barva, ale nejsou nutné.

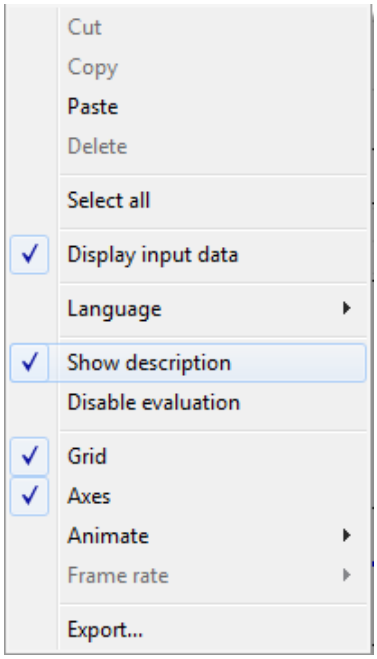


Obrázek 5.3: Zobrazení více průběhů barevně odlišených do jednoho grafu s popiskem

U každého grafu je možnost zapnout název grafu a graf si podle potřeby pro přehlednost pojmenovat. Pokud si vybereme daný graf a klikneme na něj pravým tlačítkem myši, získáme několik možností (platí pouze pro výchozí graf ve SMath studiu, nikoliv plugin, tam už pracujeme trochu jinak, ale obdobně):

- **Displey input data** nám buď zobrazí, nebo skryje zdroj hodnot (v obrázku 5.3 je to $f(x)$ dole pod grafem).
- **Show description** umožňuje zobrazit nebo skrýt možnost pojmenování grafu (v horní části grafu).
- **Grid** a **Axes** nám umožní zobrazit nebo skrýt mřížku či osy grafu.
- **Export** nám umožní daný graf převést do formy obrázku a uložit si ho samostatně někde na disk. Vhodné například pro vytváření protokolů, kde potřebujeme vložit daný graf ve formě obrázku. Není zde potřeba si graf ukládat ve formě vyfoceného obrázku obrazovky, ale uděláme to tedy mnohem jednodušeji a přesněji. Ulehčí nám to práci s grafy.

V grafu si můžeme zobrazit průběh pouze v rozmezí, které si sami definujeme. Pokud by nás zajímala jen nějaká část, můžeme použít například možnost bloku if – else. V části Programming můžeme vybrat z více možností různých cyklů (jedná se zde o „programátorskou část“, kterou můžeme využít). V našem případě si ukážeme vykreslení funkce cosinus pouze, pokud se hodnota x bude pohybovat od π do $5 \cdot \pi$. Pochopitelně můžeme si zvolit i jiné možnosti. SMath studio v tomto nabízí zajímavé možnosti.



Obrázek 5.4: Možnost práce se zobrazením grafu

5.2 3D grafy

Ve SMath studiu je možnost vkládat i 3D grafy. Tedy grafy, které jsou prostorové a potřebují k zrekonstruování (sestavení) tři hodnoty. Hodnoty na $ose\ x$, $ose\ y$ a $ose\ z$. Pokud chceme do listu vložit takový graf, postupujeme stejně jako v případě 2D grafu.

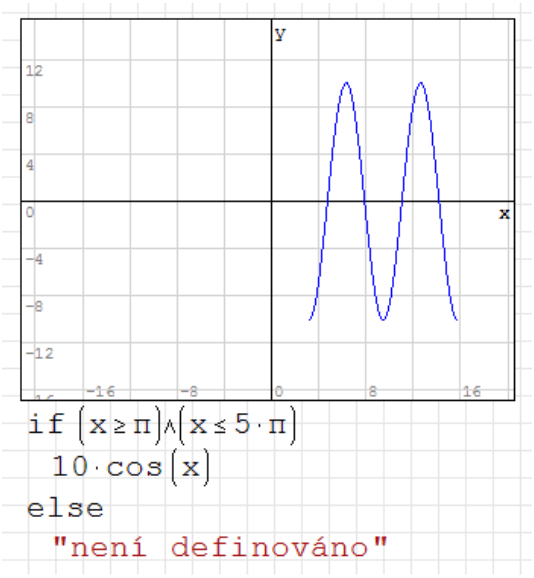
Zvolíme Insert -> Plot -> 3D.

V nabídce Plot pak můžeme s takovým grafem pracovat. Je zde možnost s grafem otáčet do různých směrů (rotovat), zvětšovat nebo zmenšovat zobrazení (pomocí Scale), posouvat, ale také zobrazit si graf pouze jako body, anebo spojený linkami, což je základní nastavení.

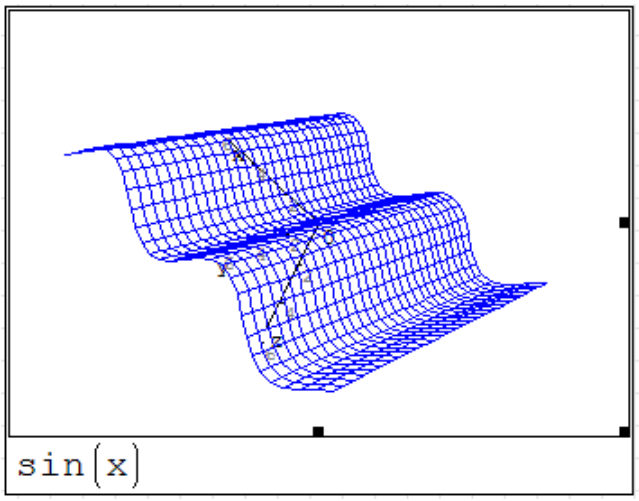
Dále pracujeme s grafem obdobně jako s 2D grafem. Rozdíl je v zobrazení a v počtu souřadnic, které potřebujeme. Pro vytvoření 3D grafu potřebujeme tři souřadnice, jelikož se pohybujeme ve 3D prostoru. Potřebujeme souřadnici x, y a z. Více v obrázku, kde je zobrazeno vykreslení trojúhelníku ve 3D prostoru.

Druh grafu	počet souřadnic	souřadnice
2D	2	x, y
3D	3	x, y, z

Tabulka 5.1: Tabulka grafy, souřadnice



Obrázek 5.5: Zobrazení určité části průběhu v grafu



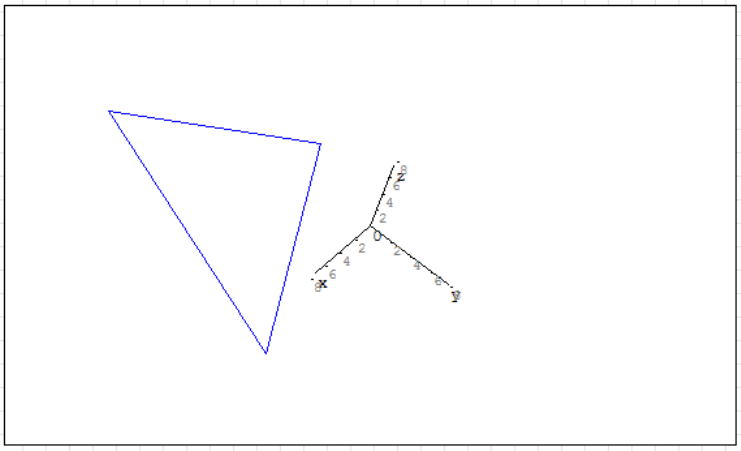
Obrázek 5.6: Jednoduchý 3D graf

Obrázek 5.7: Nabídka plot pro práci s grafem

$$B := \begin{pmatrix} 12 & 20 & 11 \\ 12 & 12 & 15 \\ 20 & 11 & 11 \\ 12 & 20 & 11 \end{pmatrix}$$

Graf 3D potřebuje tři souřadnice, x, y a z

Zde je vytvořený jednoduchý 3D graf pomocí matice 4x3



Obrázek 5.8: Vytvoření 3D grafu maticí

6 Vytvoření a popis výukových scriptů

V praktické části bakalářské práce jsem vytvořil návrh IIR filtrů (filtry s nekonečnou impulzní odezvou). Dá se velmi dobře použít k lepšímu pochopení učební látky na toto téma, popřípadě se tímto můžete inspirovat a navrhovat další řešení.

Ve scriptu, který je návrhem IIR filtru jsem pro větší přehlednost použil několik barevných značení. Můžete tak podle barvy poznat, kterou veličinu můžete zadat, který se vypočítá apod. Vybral jsem toto barevné značení:

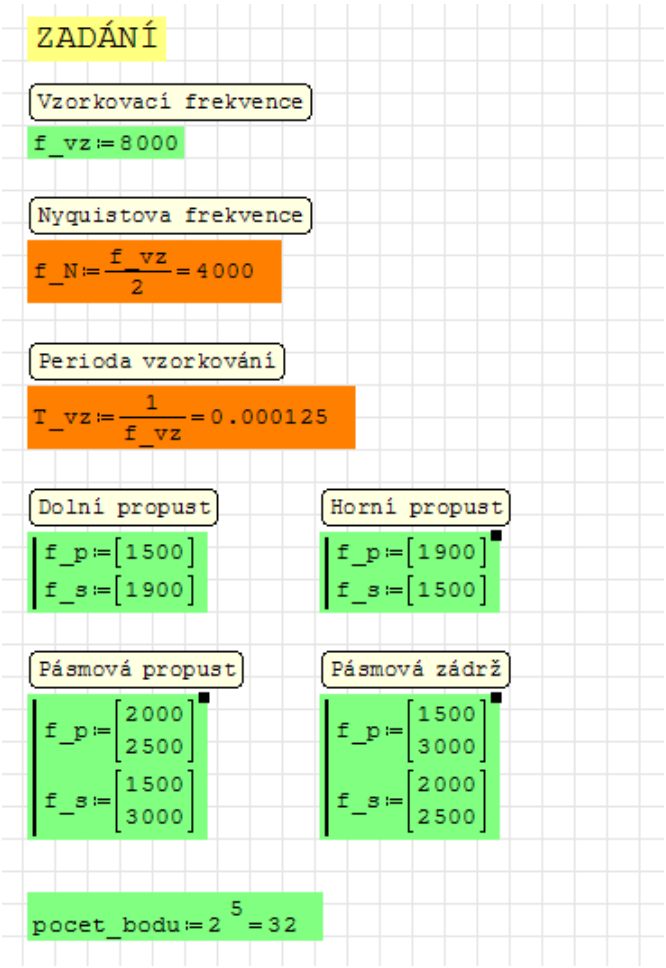
- **Žluté označení** - značí komentáře, popisky a nadpisy
- **Zelené označení** - značí zadání (zde si veličinu volí uživatel a sám dosazuje)
- **Oranžové označení** - jedná se o mezivýsledky a výsledky výpočtů
- **Šedé označení** - označuje jednotlivé funkční bloky (např. cykly, funkci line apod.)
- **Růžové označení** - toto označení ve scriptu sice nenajdete, ale uvádím ho zde jako příklad toho, jak si můžete značit části kódu s nějakou chybou při odladʼování

Na obrázku 6.1 vidíme první část návrhu filtru. Zadáme vzorkovací frekvenci, která musí být minimálně dvojnásobkem Nyquistovi frekvence. V této části scriptu jsem Nyquistovu frekvenci získal výpočtem, když jsem vzorkovací frekvenci podělil dvěma. Ze vzorkovací frekvence pak získáme periodu vzorkování, a to převrácením hodnoty vzorkovací frekvence. Dále musíme zadat frekvence filtru. Zde je potřeba si zároveň vybrat, který filtr chceme navrhovat. V případě, že si vybereme například filtr typu pásmová zádrž, musíme ostatní tři varianty vynechat z návrhu (to uděláte tak, že na proměnné aplikujeme tzv. disable evalution). V prostředí se nám pak tyto proměnné označí černým čtverečkem vpravo nahoře vedle danné proměnné. Zvolil jsem zde sloupcové vektory, jelikož se s nimi pracuje jako z výchozí variantou SMath studia.

U zadání si můžeme vybrat dvě varianty. Můžeme zadat přípustný pokles magnitudy v propustném a nepropustném pásmu v dB a dopočítáme si hodnoty A_p a A_s. Na druhou stranu, ale můžeme zadat i hodnoty A_p a A_s, kde bude výsledkem přípustný pokles magnitudy v propustném a nepropustném směru v dB. U této části zadání musíme vybrat pouze jednu z variant, pokud se rozhodneme pro zadání v dB, musíme druhou část zadání z výpočtu vyřadit. Na obrázku jsou takto takové výpočty označeny černým čtverečkem v pravém horním rohu. Uděláme to jednoduchým způsobem, označíme vzorce, které chceme vynechat a pod položkou EDIT zvolíme Disable evalution.

Důležitou částí je i volba typu aproximace, kterou aplikujeme. Můžeme si vybrat ze tří možností:

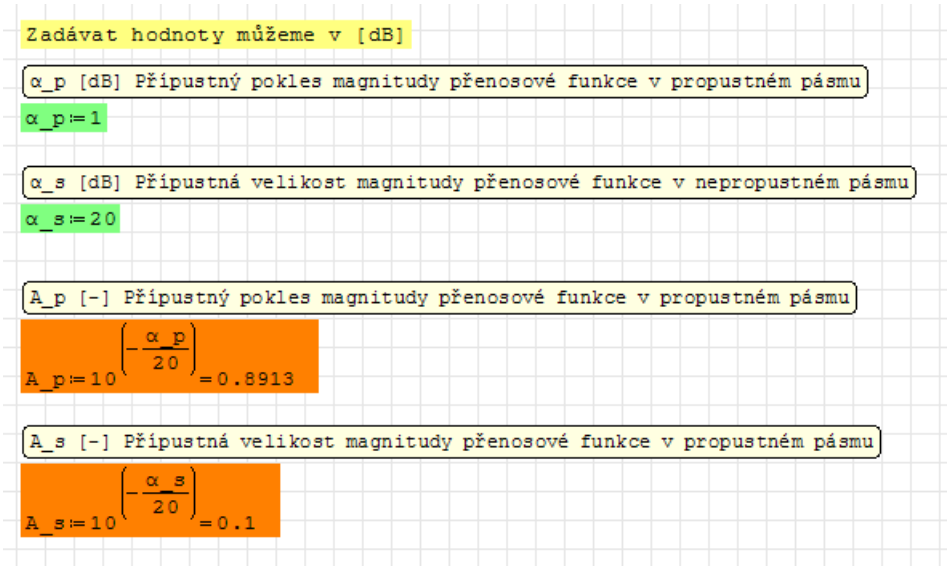
- Volba approx:="butter" - Butterworthova aproximace
- Volba approx:="cheby1" - Čebyševova aproximace 1. druhu
- Volba approx:="cheby2" - Čebyševova aproximace 2. druhu (inverzní)



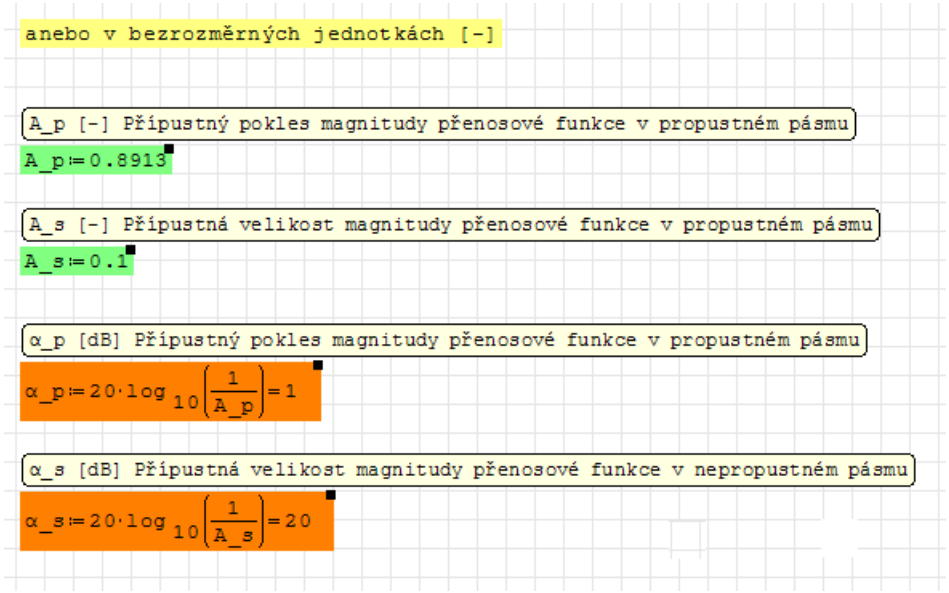
Obrázek 6.1: Zadání vstupních parametrů IIR filtru

V další části návrhu IIR filtru si určíme typ filtru, který se bude navrhovat, což vidíme na obrázku 6.5. Použil jsem programovací sekvenci if - else. Nejprve si zjistíme velikost dimenze frekvencí f_p a f_s. Jesliže je dimenze rovna jedné víme, že se bude jednat o horní nebo dolní propust. Upřesnění propusti zjistíme snadno. Pokud je hodnota frekvence f_s větší než hodnota frekvence $f_p(f_s > f_p)$ jedná se o dolní prospust, v opačném případě se jedná o horní propust. V tomto kroku přiřadím výsledek do proměnné typ_filtru. V opačném případě, pokud nám vyjde, že dimenze proměnné f_p a f_s je větší, jedná se o pásmovou propust nebo pásmovou zádrž. V případě, že hodnota frekvence f_s s indexem jedna je menší než hodnota frekvence f_p s indexem jedna a zároveň je hodnota frekvence f_p s indexem dva menší než hodnota frekvence f_s s indexem dva, tak se jedná o pásmovou propust, opačný případ značí pásmovou zádrž.

Než přejdeme z digitální do analogové oblasti, tak si vykreslíme toleranční schéma námi zadaného filtru. Na obrázku 6.6 vidíme ukázku tolerančního pásma pro filtr typu

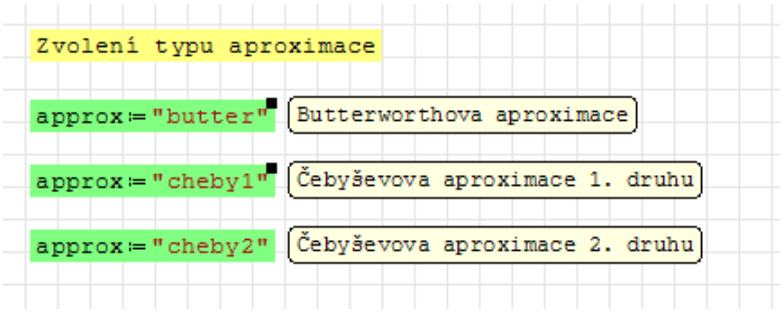


Obrázek 6.2: Zadání v dB

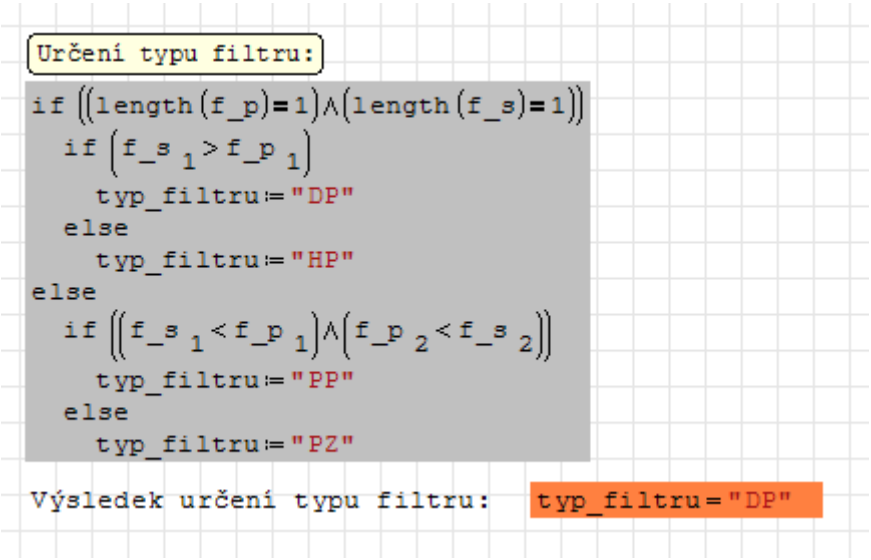


Obrázek 6.3: Zadání s výpočtem v dB

dolní propust. Do pomocných proměnných $plotDP$ a $plotHP$ $plotPP$ a $plotPZ$si uložíme body, které nám pak ve výsledném grafu vytvoří linii pro zobrazení pásma. Pomocné proměnné si vytvoříme tak, že zadáme hodnoty f_s a f_p vždy pro hodnotu v bodě 0 a pro hodnotu v bodě 1. Pokud se jedná o filtr typu pásmová zádrž nebo pásmová propust, tak se ve tvorbě liní objeví ještě další dvě přímky. Je to způsobeno tím, že velikost dimenze těchto filtrů je rovna dvěma. Další dvě linie vytvoříme po zadání hodnoty A_p - přípustný



Obrázek 6.4: Volba aproximace



Obrázek 6.5: Určení typu filtru

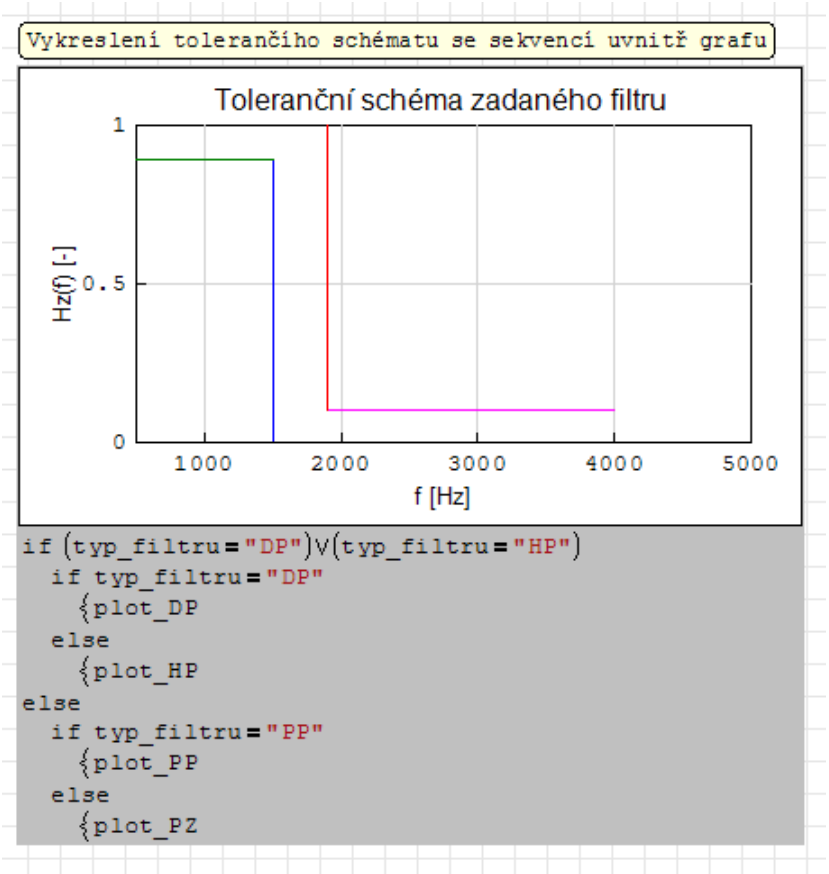
pokles magnitudy přenosové funkce v propustném směru a hodnoty A_s - přípustné velikosti magnitudy přenosové funkce v nepropustném směru. Pro správné vykreslení tolerančního schématu už nám stačí jen v rozhodovací sekci grafu porovnat jestli se jedná o dolní nebo horní propust, pásmovou zádrž nebo propust. Do vysledného X Y grafu se pak vloží správná varianta pomocné proměnné, kterou jsme si vysvětlili výše tohoto odstavce.

Přípustný pokles magnitudy přenosové funkce v propustném směru:

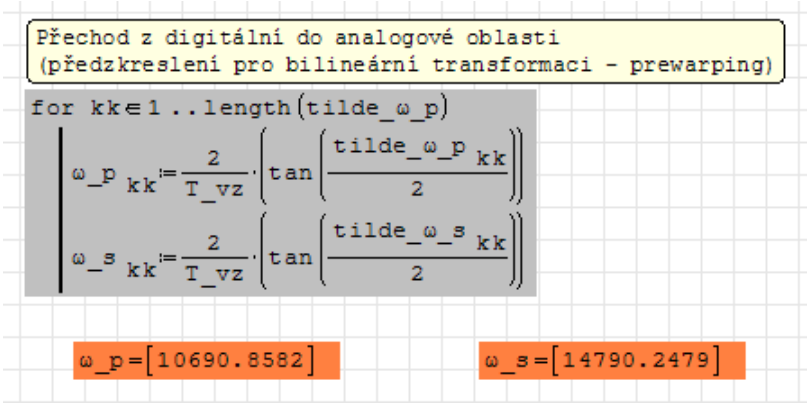
$$A_p = 10^{-\frac{1}{20}} = 0,8913 \tag{6.1}$$

Přípustná velikost magnitudy přenosové funkce v nepropustném směru:

$$A_s = 10^{-\frac{20}{20}} = 0,1 \tag{6.2}$$



Obrázek 6.6: Zobrazení tolerančního pásma DP



Obrázek 6.7: Přechod z digitální do analogové oblasti

Na začátku jsme si zvolili typ aproximace. V další fázi návrhu IIR filtrů se provede v sekci **if**–**else** právě Vámi vybraná varianta. Na obrázku 6.8 vidíme čebyševovu aproxi-

maci 2. řádu, tedy inverzní. V závěru varainty čebyševovi inverzní aproximace je použita funkce stack. Tato funkce nám zajistí to, že se dva sloupcové vektory seřadí pod sebe do jednoho sloupcového vektoru. Opačným případem je pak funkce augment, ta řadí dva vektory vedle sebe. Postup lze vidět i v následujících vzorcích, které jsem čerpal z interních materiálů katedry 1.

$$N_p = \frac{acosh\frac{1}{k_1}}{acosh\frac{1}{k_0}} \tag{6.3}$$

$$\alpha_{ell} = \frac{\left(\sqrt{1+\frac{\epsilon^2}{k_1^2}}+\frac{\epsilon}{k_1}\right)^{\frac{1}{N_p}} - \left(\sqrt{1+\frac{\epsilon^2}{k_1^2}}-\frac{\epsilon}{k_1}\right)^{\frac{1}{N_p}}}{\frac{1}{N_p}^2} \tag{6.4}$$

$$\beta_{ell} = \frac{\left(\sqrt{1+\frac{\epsilon^2}{k_1^2}}+\frac{\epsilon}{k_1}\right)^{\frac{1}{N_p}} + \left(\sqrt{1+\frac{\epsilon^2}{k_1^2}}-\frac{\epsilon}{k_1}\right)^{\frac{1}{N_p}}}{\frac{1}{N_p}^2} \tag{6.5}$$

$$p_{infty} = \frac{-\alpha_{ell}\cdot sin((2\cdot k-1)\cdot\frac{\pi}{2\cdot N_p}) - (i\cdot\beta_{ell}\cdot cos((2\cdot k-1)\cdot\frac{\pi}{2\cdot N_p}))}{\alpha_{ell}^2\cdot sin((2\cdot k-1)\cdot\frac{\pi}{2\cdot N_p})^2 + \beta_{ell}^2\cdot cos((2\cdot k-1)\cdot\frac{\pi}{2\cdot N_p})^2} \tag{6.6}$$

Nulové body pro N_p sudé i liché:

$$p_0 = \frac{i}{cos(\frac{(2\cdot k-1)\cdot\pi}{2\cdot N_p})} \tag{6.7}$$

Násobný koeficient bude pro N_p sudé:

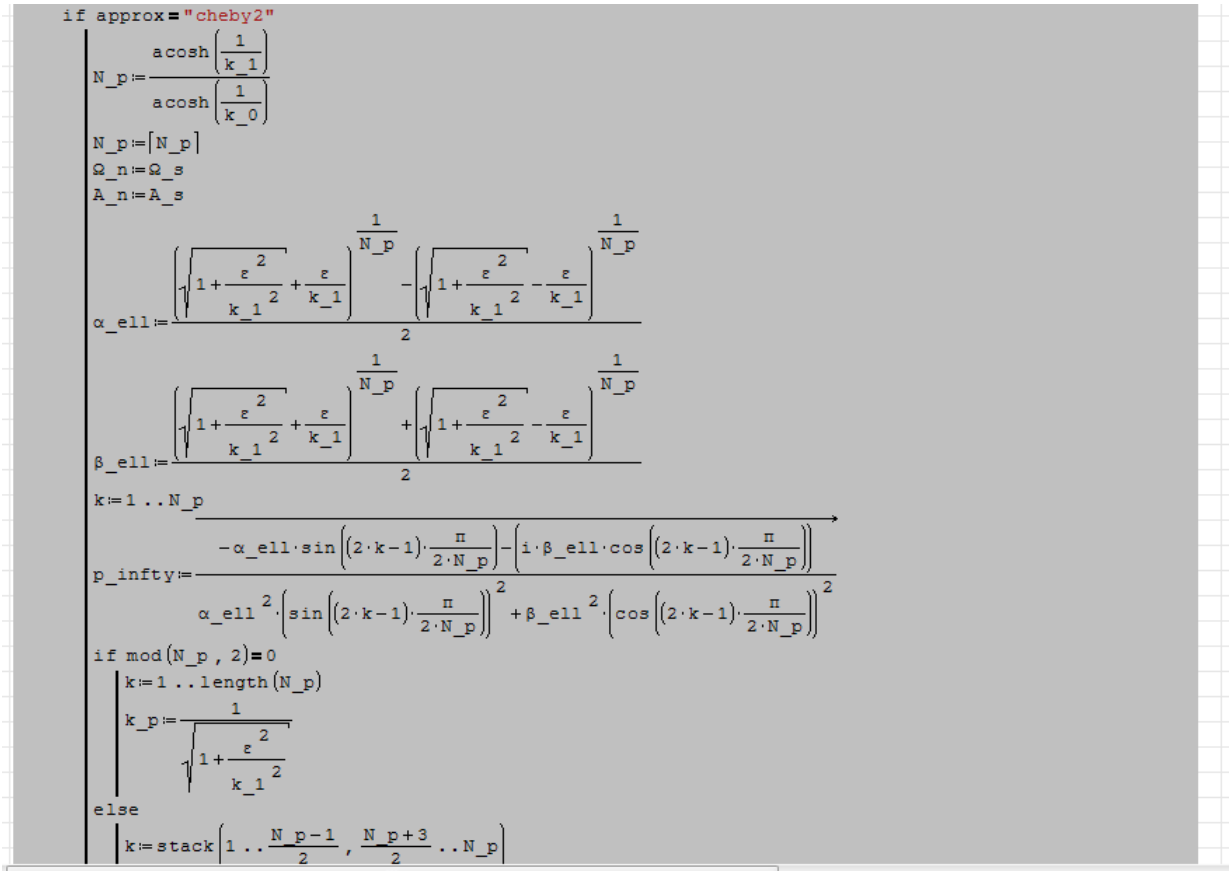
$$k_p = \frac{1}{\sqrt{1+\frac{\varepsilon^2}{k_1^2}}} \tag{6.8}$$

Násobné koeficienty pro N_p liché:

$$k_p = N_p\cdot\frac{k_1}{\varepsilon} \tag{6.9}$$

Na náhledu 6.9 se přesouváme ke koeficientům přenosové funkce $H_p(p)$. Vidíme nulové body a pro kontrolu dosadíme do přenosových funkcí právě nulové body. Výsledkem tak vždy bude vektor nul.

Pro grafické zobrazení průběhu frekvenčního přenosu $H_p(\Omega)$ potřebujeme na osu x získat hodnoty Ω. V Matlabu bychom mohli využít funkce linspace, ve SMath studiu jsem ale musel obdobnou funkci vytvořit (respektive správně vyplnit funkci range(3)). Názorně se můžeme podívat na obrázku 6.10. Pokud bychom nevytvořili vektor právě takto, jednotlivé prvky vektoru by neodpovídaly prvkům, které byste si vytvořily například v Matlabu. Je potřeba tedy správně pronásobit jednotlivé prvky. V přední části před závorkou násobíme nulou součet vektoru jedniček a daného kroku, se kterým chceme



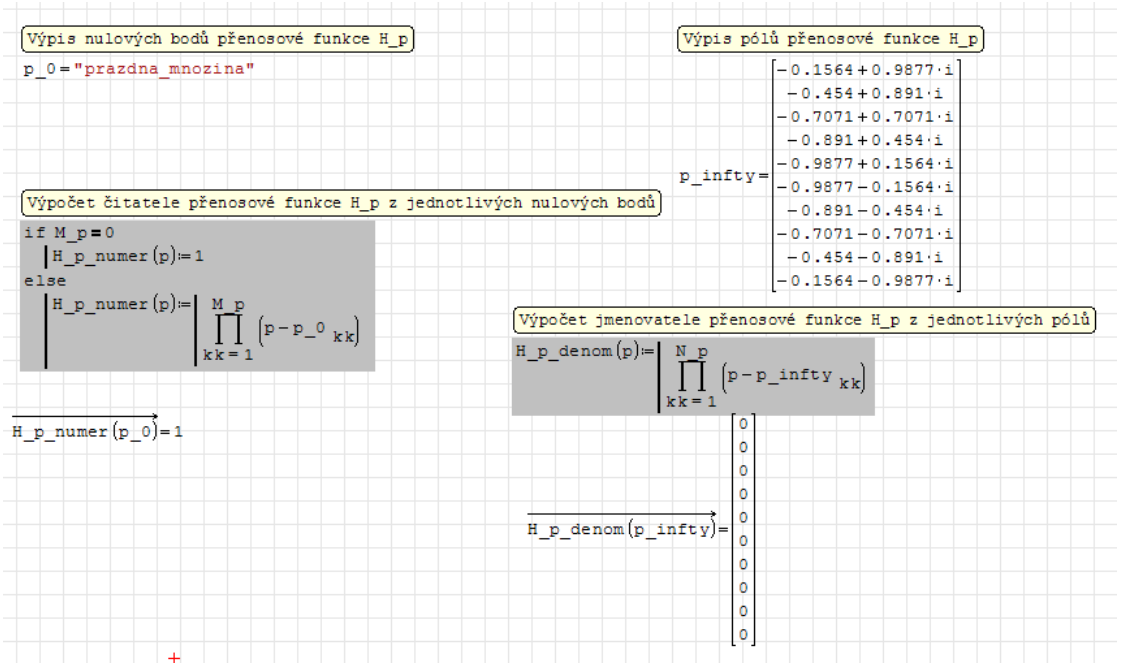
Obrázek 6.8: Chebyševova aproximace 2. řádu (inverzní) - výřez postupu

výsledný vektor vytvořit. Ve druhé části pak určíme počet jednotlivých prvků, které už budou s daným krokem z přední části funkce.

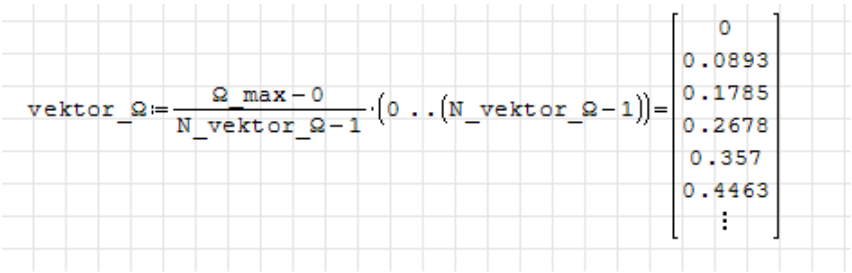
Pro zobrazení průběhu frekvenčního přenosu H_p ještě potřebujeme na osu y hodnoty této přenosové funkce v závislosti na vektor Ω. Obrázek 6.11 ukazuje právě toto řešení. Přenosovou funkci jsem si rozdělil na výpočet koeficientů čitatele a jmenovatele a následně jsem vytvořil konečnou přenosovou funkci. Obdobně se pracovalo i s ostatními přenosovými funkcemi ve scriptu. Důležité pro tuto část návrhu je i to, že se řešení dělí na dvě možnosti. Pokud máme v proměnné M_p nulu, musíme použít pro výpočet přenosové funkce jiný vzorec, ve kterém je na pozici čitatele jednička. Znamená to, že v případě aproximace podle Butterwortha a Čebyševa1 nemáme koeficienty čitatele, jak vypočítat, jelikož neexistují nulové body.

Abychom mohli graficky průběh znázornit, musíme mít připravenou proměnnou, kterou následně vložíme jako vstupní hodnotu grafu. V tomto případě je nejvhodnější řešení použít funkci augment(), která zajistí, že se oba sloupcové vektory vloží do matice vedle sebe. Viz. právě obrázek 6.12, kde je to názorně zobrazeno.

Pro názornost výstupního grafu si ukážeme, jak nám vyšel průběh přenosové funkce



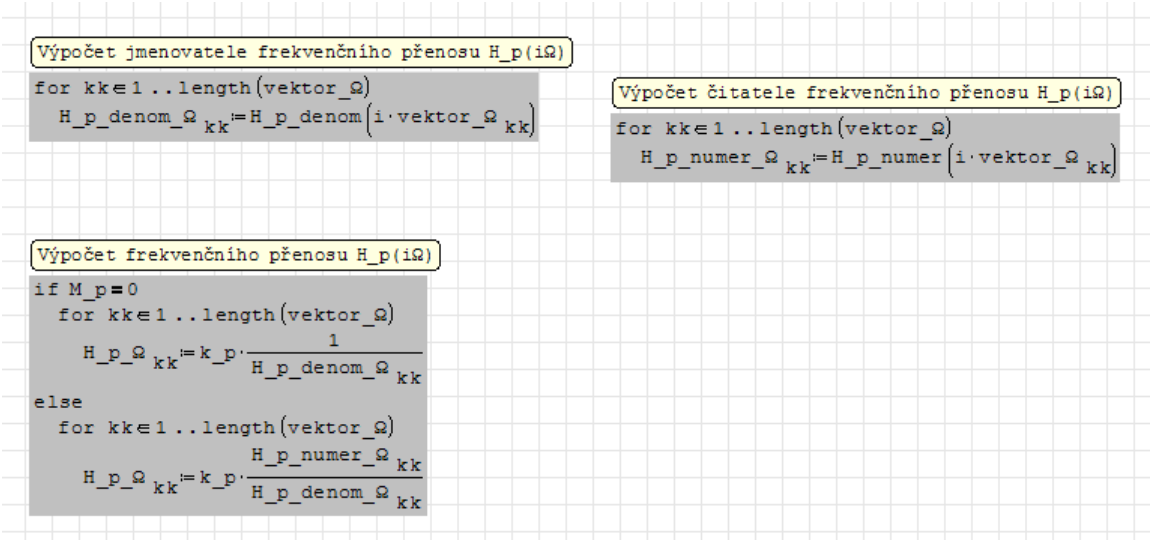
Obrázek 6.9: Výpis nulových bodů, pólů



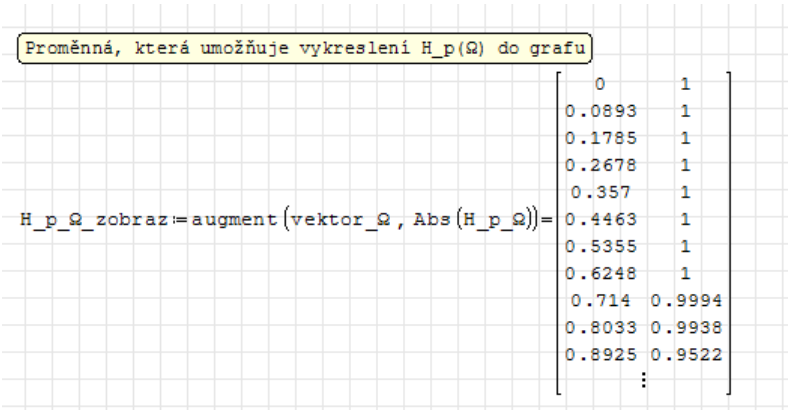
Obrázek 6.10: Vytvoření vektoru Ω

$H_p(\Omega)$ pro zadání horní propust a Čebyševovu aproximaci. Průběh vidíme na obrázku 6.13.

Dalším krokem návrhu IIR filtru byla zpětná frekvenční transformace NDP (normované dolní propusti) na propust vybraného typu. V našem případě si v těchto textech vyjimečně ukážeme transformaci na horní propust (v ostatních případech jde o zadání s dolní propustí), jelikož ta je co se týká postupu výpočtu nejzajímavější. Náhled 6.14 ukazuje právě výtažek z výukového scriptu, který řeší právě případ horní propusti. V návrhu musíme zohlednit situaci, kdy má být v čitateli pouze jednička. To je pro případ butterworthovi a čebyševovi aproximace. U čebyševovi aproximace 2. řádu, to už neplatí a v čitateli máme koeficienty odlišné jedničce. V této části také vidíme využití funkce stack(). Tato funkce nám zajistí, že se oba sloupcové vektory nesrovnají vedle sebe, ale



Obrázek 6.11: Výpočet hodnot přenosové funkce H_p



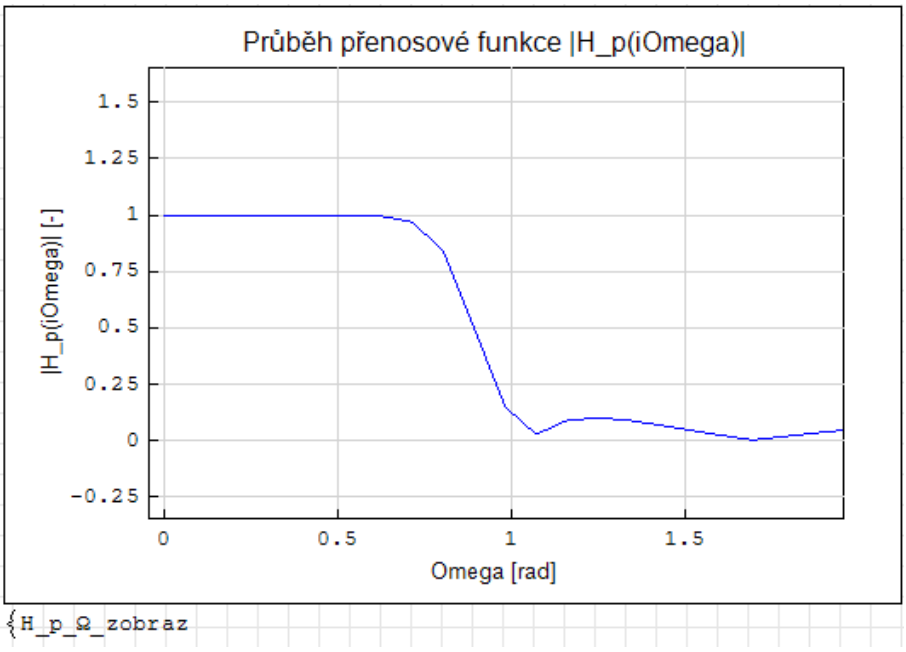
Obrázek 6.12: Proměnná pro zobrazení průběhu přenosové funkce H_p

pod sebe. Dostaneme tak jeden slopcový vektor, kde bude právě jeden sloupec.

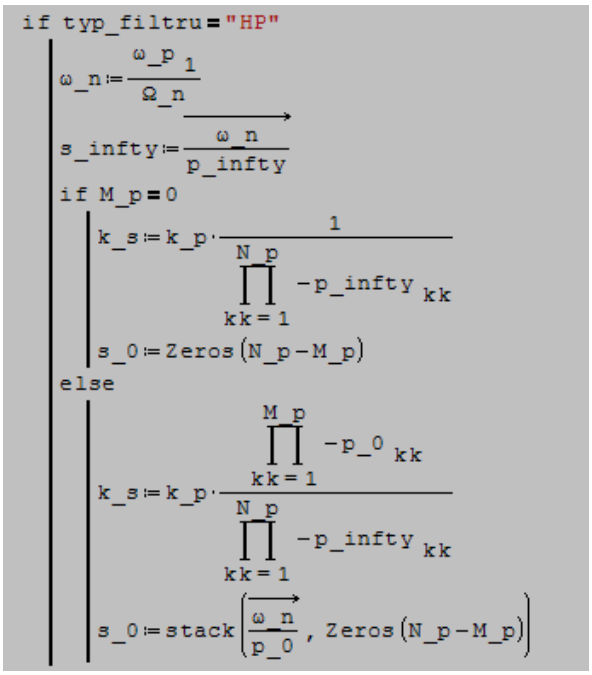
Stejně tak, pokud si chceme zobrazit průběh frekvenčního přenosu $H_s(\omega)$, musíme si vytvořit vektor ω na osu x.

Na obrázku 6.15 vidíme průběh přenosové funkce $H_s(\omega)$ pro zadání horní propust a Čebyševovu aproximaci. Zde si ukážeme, jak si graf správně nastavit, abychom viděli výstupní průběh funkce. Na obrázku 6.16 vidíme okno, které se nám zobrazí, pokud na daný graf dvakrát klikneme levým tlačítkem myši. Ve zobrazené nabídce můžeme upravovat rozsahy na osách (tak, aby byl vidět celý průběh), názvy os, název grafu a další doplňující varianty. Vše je poměrně intuitivní a po pár kliknutí zjistíte, jak se kde, co dá nastavit.

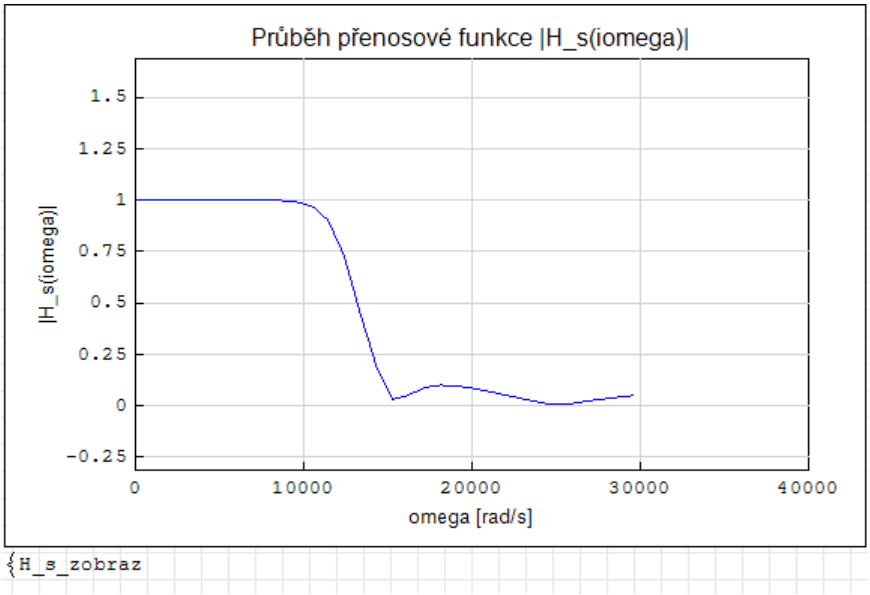
Pokud si budete chtít nastavit rozsah na osách, tak u pluginu je to velmi snadné. Po



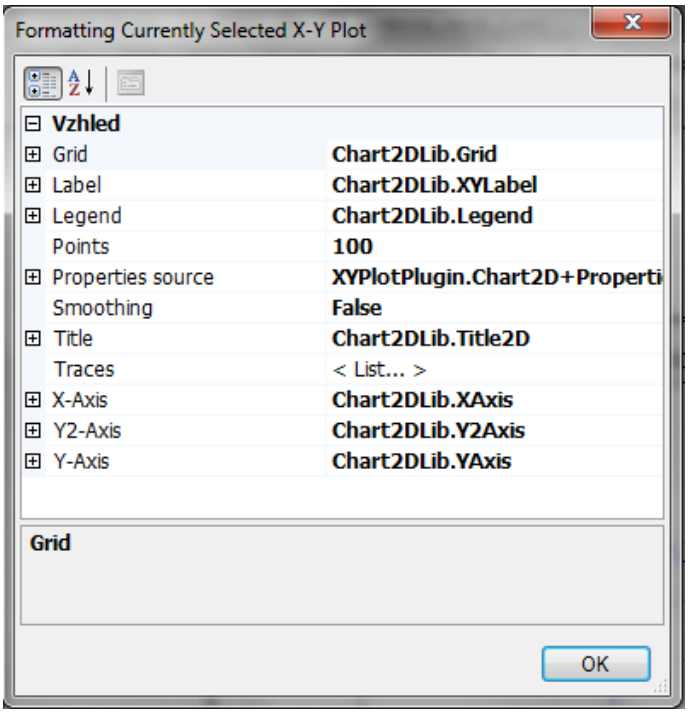
Obrázek 6.13: Zobrazení průběhu frekvenčního přenosu H_p



Obrázek 6.14: Zpětná frekvenční transformace NDP -> HP



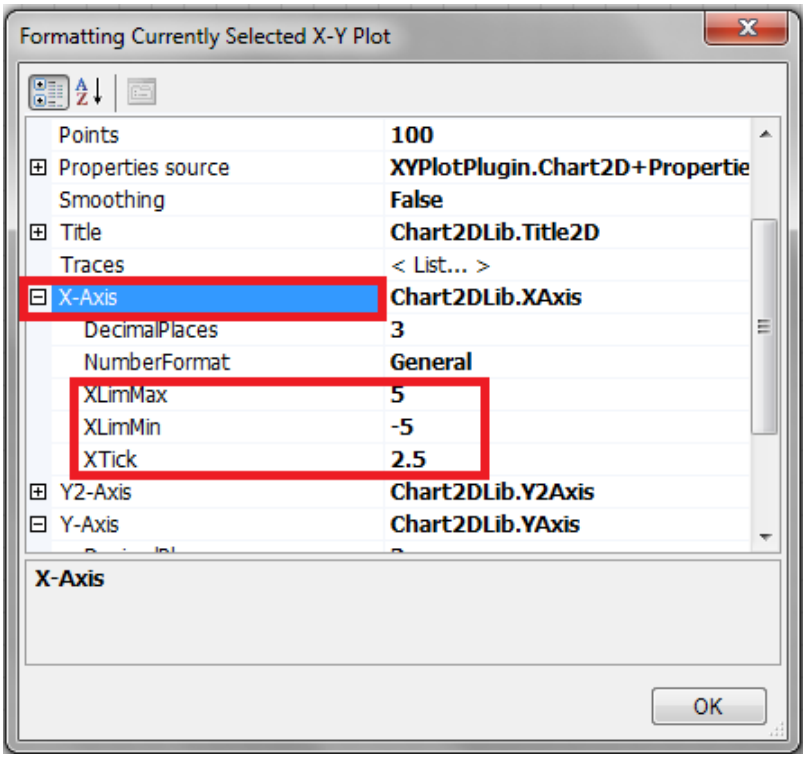
Obrázek 6.15: Zobrazení frekvenčního přenosu H_s



Obrázek 6.16: Možnosti nastavení XY grafu

dvojkliku na vybraný graf se zobrazí nabídka, kde si otevřeme položku X nebo Y osa, ná-

sledně (označeno v červeném rámečku na obrázku 6.17) si můžeme nastavit maximální hodnotu na ose a také minimální. Zároveň se dá určit i krok, se kterým se budou vypisovat jednotlivé body na ose.



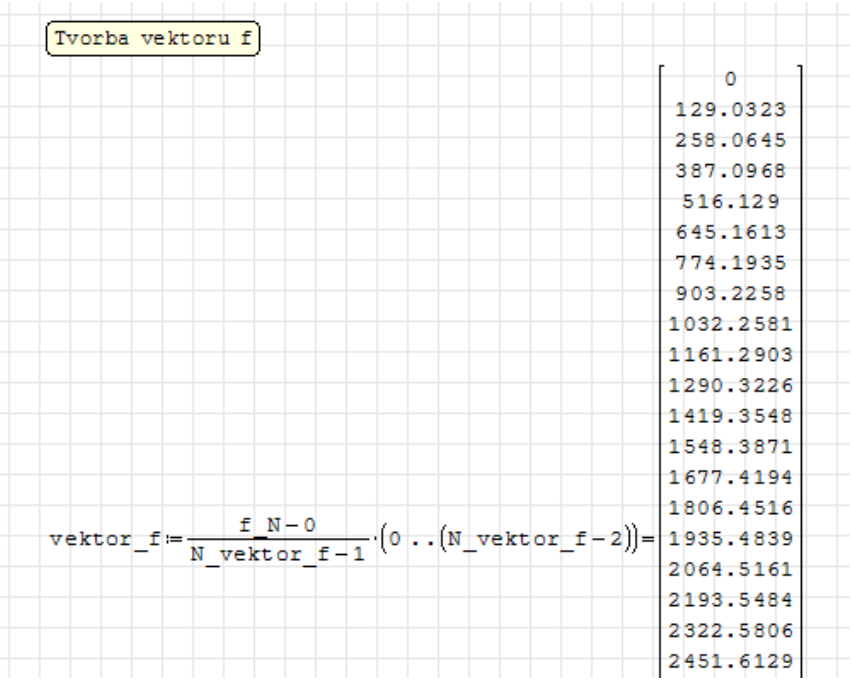
Obrázek 6.17: Nastavení rozsahu na osách X a Y

Pokud si budeme chtít zobrazit frekvenční přenos $H_z(f)$ musíme si vytvořit vektor f. Bude se jednat o hodnoty frekvencí, které použijeme na osu x. U vektoru f si můžeme v návrhu všimnout, že je použita funkce eval(). Využil jsem ji abych tak urychlil výpočet a zároveň, abych nepřeplňoval paměť počítače. Nevýhodou SMath studia je, že pracuje s hodnotami jako s analytickými výrazy, pokud chceme tuto skutečnost změnit, musíme použít právě funkci eval, která pak změní hodnoty na numerickou notaci. V případě tvorby vektoru f (na obrázku 6.18) jsme v posledním členu zvolili -2 místo -1, je to z toho důvodu, že právě poslední prvek je problematický a dochází u něho k dělení nulou, což je z matematického hlediska nesmysl.

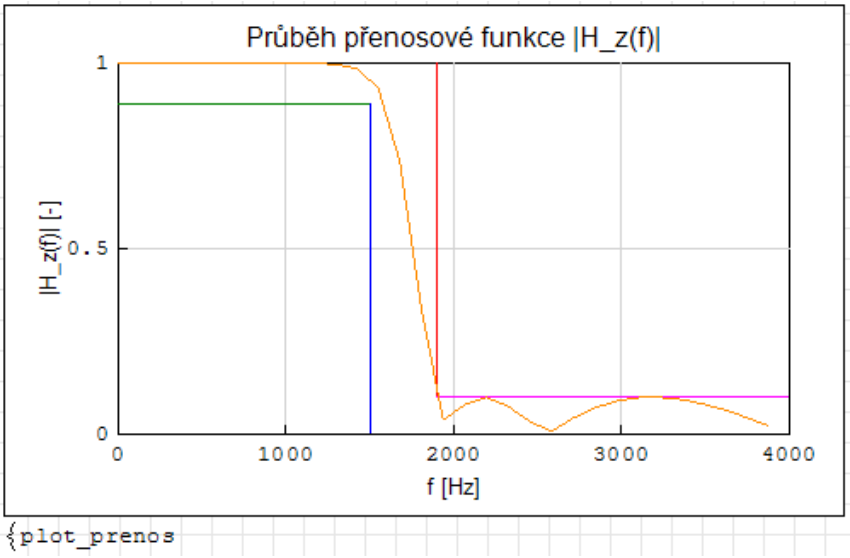
Na obrázku 6.19 si zobrazíme frekvenční přenos $H_z(f)$.

V poslední části výukového scriptu na téma IIR filtry jsem vytvořil grafické zobrazení jednotlivých nulových bodů na jednotkové kružnici. Obrázek 6.20 ukazuje předposlední krok pro grafické zobrazení. Pomocí funkce augment si vytvoříme souřadnice pólů na jednotkové kružnici (vždy reálná část a k ní hodnota imaginární části).

Pro správné zobrazení si musíme vykreslit jednotkovou kružnici. Ve SMath studiu to uděláme jednoduše pomocí funkce augment, viz. obrázek 6.21.

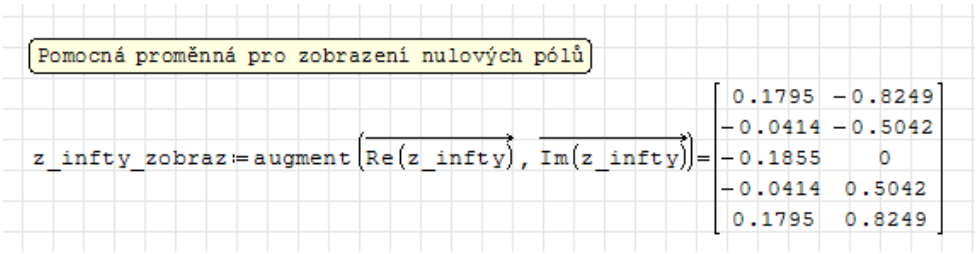


Obrázek 6.18: Tvorba vektoru f je odlišná

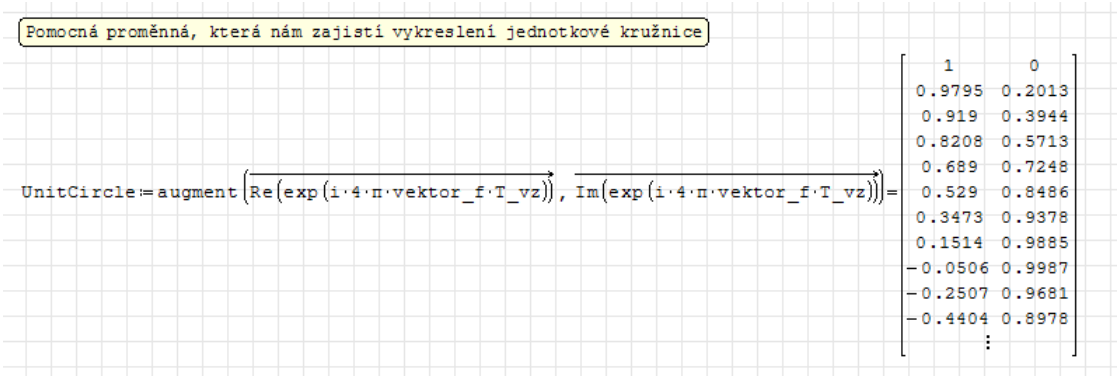


Obrázek 6.19: Zobrazení průběhu přenosové funkce $H_z(f)$

Na závěr celého výukového scriptu je zobrazení pólů přenosové funkce digitálního filtru na jednotkové kružnici. Výstupem je zobrazení na náhledu 6.23, kde vidíme výsledek pro filtr typu dolní propust a chebyševovu aproximaci. Abychom dosáhli takového



Obrázek 6.20: Pomocná proměnná pro zobrazení pólů

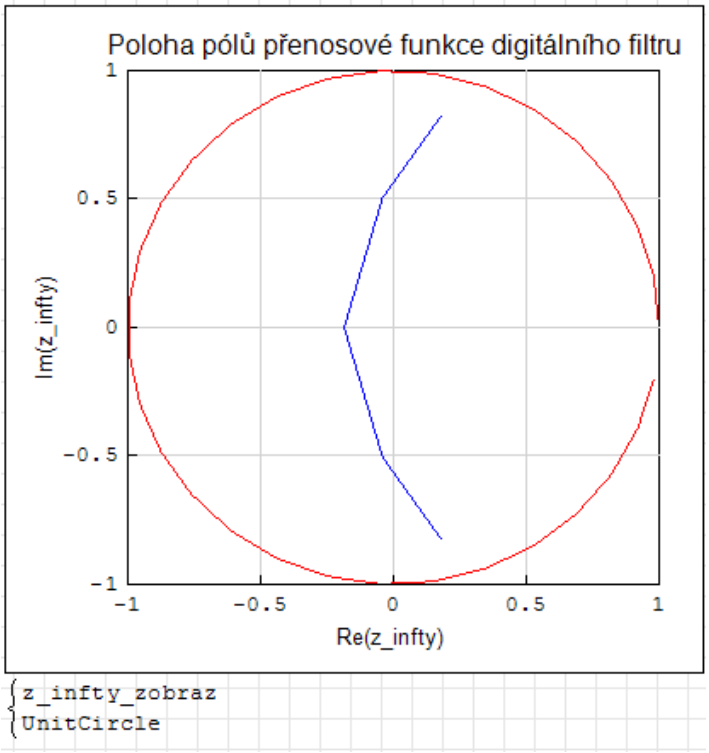


Obrázek 6.21: Tvorba jednotkové kružnice

zobrazení je potřeba správně nastavit zobrazování jednotlivých vstupních proměnných (vektorů). Před tím než si správně nastavíme graf, tak můžete vidět na obrázku 6.22, jak vypadá graf bez správného nastavení. Na obrázku můžeme vidět, že se nám jednotková kružnice vykreslila dobře, ale problém byl se zobrazením jednotlivých pólů. SMath studio primárně bere variantu Lines, což nám zajistí, že body byly spojené do jedné čáry.

Abychom graf správně nastavili, musíme na něj dvakrát kliknout. Zobrazí se nám nabídka, ve které zvolíme možnost traces. V dalším okně, které vyskočí je pak potřeba vybrat proměnnou, kterou chceme takto zobrazit a v nastavení změnit SymbolType na Vámi vybranou variantu. V našem případě to je volba Cross. Můžeme tak, ale zobrazovat i další varianty jako body, trojúhelníky a mnohem více. Zároveň si zde můžeme nastavit typ čáry, barvu čáry. Pokud bude mít výsledný graf stejný počet vstupních proměnných (vektorů apod.), tedy nebude se již měnit počet proměnných, je to skvělá možnost, jak si zde graf barevně odlišit a poukázat na jeho důležité části. V mém případě to u tohoto případu šlo. V grafu, kde zobrazuji toleranční shcéma už ne, jelikož tam byl velký rozdíl v počtu čar mezi filtry typu DP, HP a filtry PP, PZ.

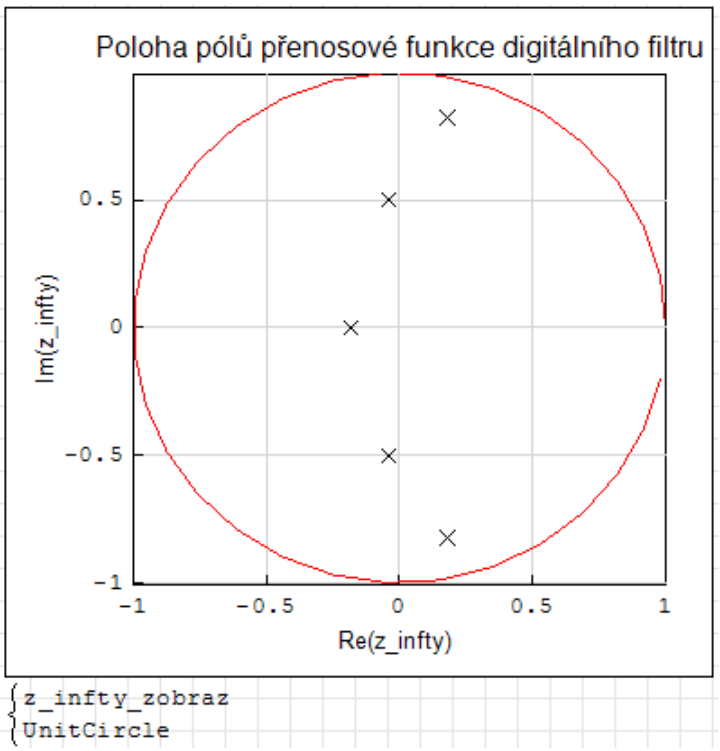
Na obrázku 6.25 se přesouváme k určení stability námi zadaného filtru. U filtrů typu IIR (filtrů s nedokonečnou impulzní odezvou) totiž může nastat problém se stabilitou. Stabilita u těchto filtrů totiž není zaručena. Nejprve si vypočítáme absolutní hodnotu pólů (v našem případě uložených v proměnné z_{infty}) daného filtru. Než provedeme výpočty vynulujeme si proměnnou $z\_infty\_geq\_0$ - zkratka qeg znamená Greater Than or



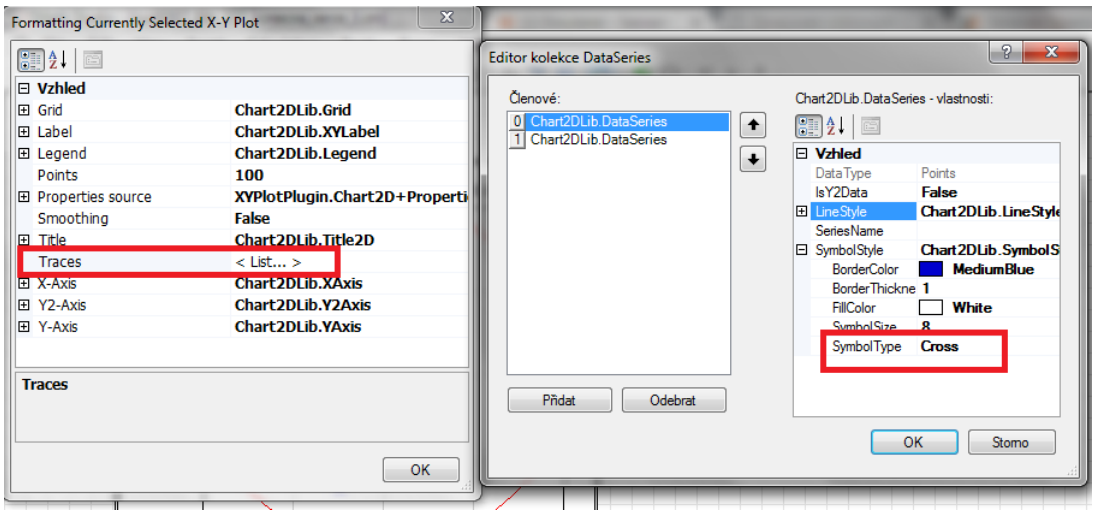
Obrázek 6.22: Zobrazení pólů přenosové funkce digitálního filtru pro DP a aproximaci cheby1 před nastavením

Equal To (v českém překladu větší nebo rovno). Následuje rozhodovací sekvence if-else, kde pokud zjistíme, že absolutní hodnota proměnné z_{infty} je větší nebo rovna 1, přičteme do proměnné $z\_infty\_geq\_0$ jedničku, v opačném případě zůstává v proměnné nula. Z teorie vyplývá, že filtr je stabilní pouze v tom případě, pokud všechny jeho póly leží uvnitř jednotkové kružnice (proto zjišťujeme jestli je hodnota pólu větší nebo rovna 1).

Konečný výsledek jestli je námi navržený filtr stabilní nebo nestabilní si zjistíme v části znázorněné na obrázku 6.26. Zjištění tohoto výsledku už je triviální záležitostí. Pouze si zjistíme v sekci if-else, co je uložené v proměnné $z\_infty\_geq\_0$. Pokud je tato proměnná rovna 0 můžeme tvrdit, že je filtr stabilní. V případě, že by byla v proměnné uložená jiná hodnota než 0 je námi navržený filtr nestabilní.



Obrázek 6.23: Zobrazení polohy pólů na jednotkové kružnici pro DP a aproximaci cheby2



Obrázek 6.24: Nastavení zobrazení výsledků jako bodů (křížků) na jednotkové kružnici

```
z_infty_geq_0 ≔ 0
for kk ∈ 1 .. length(z_infty)
  if |z_infty_kk| ≥ 1
    z_infty_geq_0 ≔ z_infty_geq_0 + 1
  else
    z_infty_geq_0 ≔ z_infty_geq_0
```

Obrázek 6.25: Pomocný výpočet pro zjištění stability filtru

```
Zjištění stability soustavy
if z_infty_geq_0 = 0
  stabilita ≔ "Všechny póly leží uvnitř jednotkové kružnice -> Filtr je stabilní"
else
  stabilita ≔ "Některý (některé) z pólů leží vně jednotkové kružnice -> Filtr NENÍ stabilní"

stabilita = "Všechny póly leží uvnitř jednotkové kružnice -> Filtr je stabilní"
```

Obrázek 6.26: Vyhodnocení stability zadaného filtru

7 Subjektivní porovnání práce v prostředí SMath studio s prostředím MATLAB

V průběhu tvorby této bakalářské práce jsem narazil na několik skutečností, které jsou mezi těmito prostředími dosti odlišné. Nabízí se porovnání obou programů. Obě prostředí nabízí velkou porci možností. Matlab má vyřešeny téměř všechny funkce pro výpočty, grafy apod. U programu SMath studio je velkou nevýhodou, že řada funkcí ještě nebyla do prostředí implementována. Na oficiálním fóru prostředí je ale celá řada typů a rad, jak tyto funkce nahradit, popřípadě řešit. V průběhu roku pak vychází několik rozšíření (balíčků), ale i přesto jsou stále některé funkce nevyřešeny. Výhoda SMath studia spočívá v tom, že je ke stažení zdarma a je možnost se podílet na jeho tvorbě a zlepšování.

Pokud se jedná o tvorbu grafů, tak mě osobně přijde Matlab mnohem více připravený. Na druhou stranu, pokud se ve SMath studiu naučíte s grafy pracovat, tak i tam můžete velmi efektivně zobrazovat průběhy funkcí, grafy výpočtů, přidávat popisky průběhů, barevné značky apod. Můžeme využít dokonce dva druhy grafů - Plot 2D/3D a X-Y graf, který je mnohem lépe editovatelný, ale musíme si jej stáhnout jako plugin.

Velkou výhodou SMath studia oproti Matlabu je zápis vzorců. Zápis matematických vztahů je svým vzhledem blízký zápisu matematických vztahů například v Latexu. Zápisy jsou mnohem lépe čitelné a jejich přehlednost je zde velkou výhodou. Navíc kostičkovaný papír působí uživatelsky velmi příjemně, výpočty a funkce můžeme navíc barevně odlišovat. Ke každému výpočtu navíc můžeme přidat i popisek, který můžeme, ale zároveň nemusíme zobrazit.

Další výhodou SMath studia oproti licencovanému Matlabu je možnost slučování výpočtů do logických celků a ještě tak více podtrhnout přehlednost zápisu. Nevýhodou těchto bloků, ale je, že nemůžeme zobrazit mezivýsledky v bloku, ale až na konci celého bloku si můžeme konečné veličiny vypsat. Zároveň zde nelze v jednom funkčním bloku použít více barev na odlišení některých částí výpočtů. V cyklech a ve funkci line nemůžeme zobrazit mezivýsledek, což koresponduje s funkčními bloky. Výhodou oproti Matlabu ale je, že pokud chceme zobrazit mezivýsledek mimo blok, stačí použít zobrazovací znaménko rovno a vidíme ihned výsledek, který je v dané proměnné. Zde si v Matlabu musíme nechat přeložit celý kód a až pak zjistíme, co v dané proměnné je. Musíte si ale dát pozor na nepříjemnou vlastnost SMath studia, kterou je přemisťování jednotlivých vztahů s výsledky. Pokud změníme například dimenzi matice, program si výsledek může přesunout do jiné části kódu a script tak nemusí vůbec fungovat, jelikož se s výsledkem dané části počítá už v následujícím bloku (jednoduše řečeno, počítá se s něčím, co ještě nemáme a teprve přijde).

Na závěr mohu konstatovat, že po uživatelské přívětivosti prostředí je na tom pro mě mnohem lépe SMath studio. Jasnou výhodou je přehlednost a možnost barevného odlišení jednotlivých bloků, vztahů, grafů apod. Můžeme tak tvořit složité scripty, ale budou pro uživatele přehledné, jelikož se vše dá dobře označit. Na druhou stranu z pohledu praktické stránky je na tom lépe Matlab, kde není potřeba většinu funkcí vymýšlet, jelikož už existují.

Nevýhodou SMath studia může být jeho rychlost výpočtů. Tím, že se jednotlivé výpočty na stránce provádí vždy znovu celé, můžeme si někdy u některých složitějších funkcí na výsledek pěkně počkat. U Matlabu byla práce v tomto směru mnohem rychlejší. Částečným řešení je funkce `eval()`, která ve SMath studiu je, ale ani to vždy nestačí.

8 Závěr

V této bakalářské práci jsem uživatele seznámil s prací ve volně stažitelném prostředí SMath studio. Uživatelé se tak mohu naučit základy v tomto programu, podívat se na řešení v praxi (vytvořil jsem výukový script na téma IIR filtry). Cíle této práce byly naplněny, jelikož tento program mohou využívat například studenti v předmětech se zpracováním signálů, mohou tak lépe a rychleji pochopit danou problematiku probírané látky. Program si může stáhnout každý, jedná se o freeware. Cílem bylo zároveň poukázat na úskalí při tvorbě v tomto programu, vše bylo dokumentováno nejen v samostatných souborech (examplech), ale také přímo v ukázce tvorby IIR filtru.

V závěru bych chtěl ještě zdůraznit, že jsem použil větší počet názorných obrázků s popisky, jelikož program SMath studio neumožňuje výpis kódu tak jako je to v jiných programech. Přepisovat jednotlivé vztahy do práce by bylo zbytečné, jelikož jak už jsem zmínil v části, kde srovnávám SMath studio a Matlab, tak zápis matematických vztahů je velmi přehledný a velmi se blíží zápis matematických vztahů například v Latexu, ve kterém je práce vytvořena.

V průběhu práce se vyskytlo několik problémů, které jsem musel řešit. Poměrně dlouhou dobu mi trvalo například pochopení práce s funkcí vectorize, která má usnadnit práci s proměnnými typu vektor a matice. Zjistil jsem, že je tato funkce poměrně nespolehlivá, jelikož to, co Vám může fungovat na jednom řádku, nemusí fungovat v analogickém vzorci na řádku dalším. V práci jsem tak tuto funkci využil, ale několik částí jsem musel řešit klasicky přes cykly.

SMath studio tak nabízí studentům dobrou alternativu k licencovaným programům. Nevýhoda je, ale v tom, že ve SMath studiu ještě nebyla řada funkcí implementována a někdy se musí složitě vytvářet. Na druhou stranu se program neustále vyvíjí a funkce přibývají.

Josef Nudera

9 Reference

[1] J. Skapa: *Zpracování číslicových signálů*. Interní materiály katedry telekomunikační techniky, Ostrava, 2016.

[2] A. Ivashov: *Oficiální webová stránka programu SMath studio* . [online]. [cit. 2017 - 04 -23].
http://en.smath.info/view/SMathStudio/summary

[3] Bernard Liengme: *SMathPrimer*. [online]. [cit.2017-04-23].
http://smath.info/?file=739837

[4] *Webová stránka SOFTONIC INTERNACIONAL S.A.*. (1997 - 2016).
https://smath-studio.en.softonic.com/#app-softonic-review

[5] *SMath studio fórum - stažení pluginu XY graph*.
http://en.smath.info/forum/yaf_poststt1774_X-Y-Plot-Region.aspx

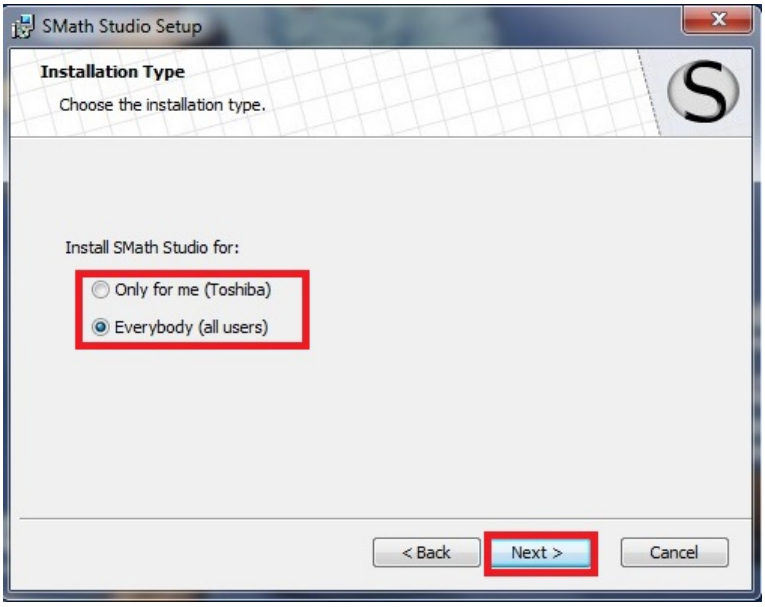
[6] *SMath studio fórum - butterworth*.
Vlozeno na DVD

A Instalace SMath studia pod operačním systémem Windows

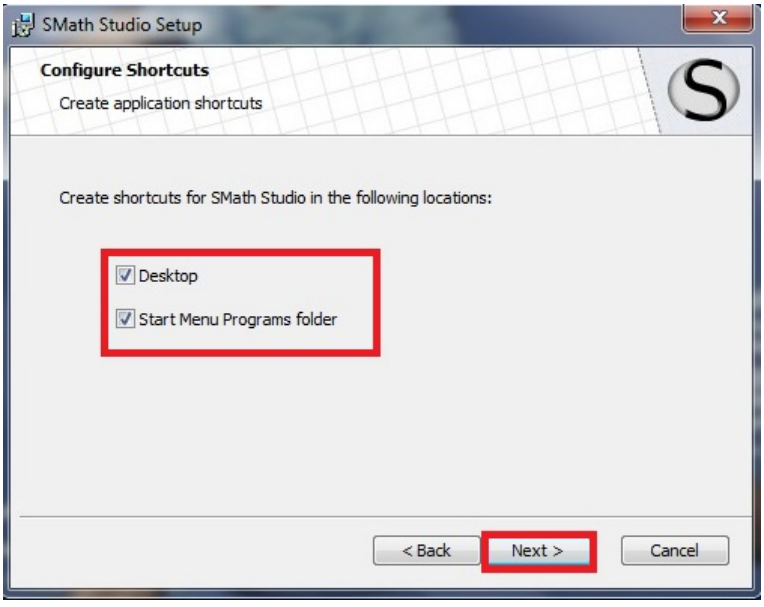
Pro jednoduchost instalace jsem obrázkový přehled instalačních kroků pod systémem Wimdows vložil do této přílohy.

Obrázek A.1: Nabídka Wizard

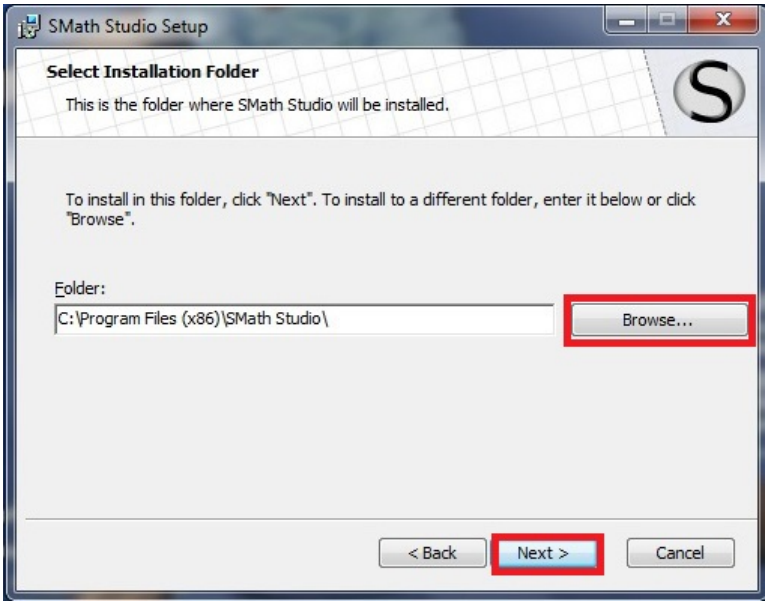
Obrázek A.2: Licenční podmínky



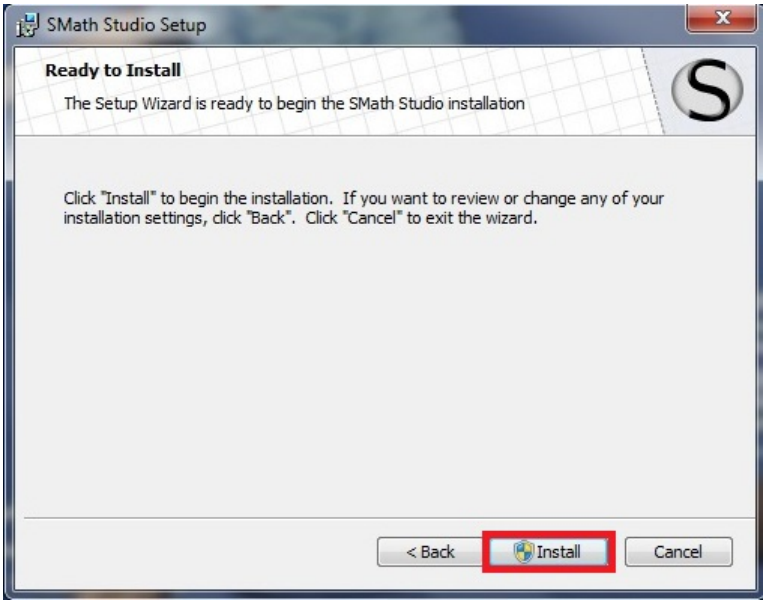
Obrázek A.3: Možnost instalace pro uživatele



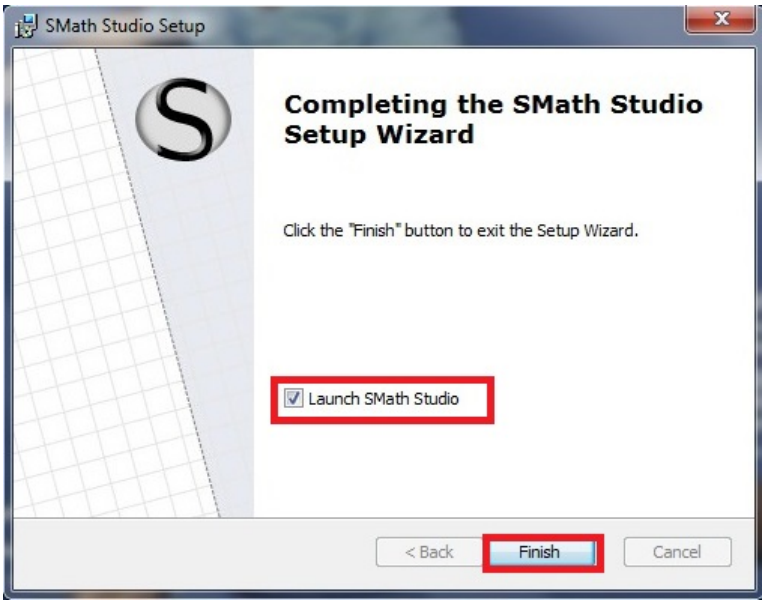
Obrázek A.4: Vytvoření zástupce



Obrázek A.5: Výběr místa na disku k instalaci



Obrázek A.6: Spuštění instalace



Obrázek A.7: Dokončení instalace